EXTRACCIÓN DE CARACTERÍSTICAS DE TEXTOS Y CLASIFICACIÓN SEGÚN GÉNERO LITERARIO MEDIANTE REDES NEURONALES

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

Autor: Pablo León Pacheco
Tutor: Ángel García Crespo

miércoles, 21 de junio de 2017

Resumen

Los géneros literarios son una de las muchas maneras de clasificación utilizadas para separar las obras literarias. Para ello se pueden emplear diversos métodos de clasificación, entre los cuales las Redes de Neuronas Artificiales son las investigadas en este proyecto. A lo largo del documento analizaremos el alcance del proyecto de investigación, cuyo fin es comparar los modelos con base en redes neuronales artificiales que ofrecen los distintos proveedores de soluciones de Aprendizaje Automático en la nube, seleccionando los que se consideren aptos para realizar dicha tarea de clasificación. Se detallarán los requisitos para la realización de cada prueba, y se analizarán los resultados obtenidos en modo comparativo para alcanzar una conclusión final.

Abstract

Literary genres are one of many classification categories employed to separate literary works. To that end there are several and diverse classification methods, among which Artificial Neural Networks are the ones researched in this project. Throughout the document we will analyse the research project span, which aims to compare the different models based on artificial neural networks which are provided by different solutions of cloud based Machine Learning, selecting those considered suitable to carry out said classification task. The requirements for carrying out each test will be detailed, and the results obtained will be compared to arrive at a conclusion.

Agradecimientos

Ante todo, agradecer a mis padres por todo el apoyo y fe que han tenido en mí durante estos 4 años, no habría tenido acceso a estos estudios y realmente a nada sin su incondicional apoyo y cariño.

A mi madre especialmente por aguantar tantas veces mi estrés irracional, mis enfados por una compilación errónea y mis rabietas porque la nota no era la que merecía, y seguir apoyándome con todo pese a ello. Gracias por tus consejos, aunque siempre los desprecie, nunca los olvido.

A mi padre por compartir esta pasión que es la informática, por darme consejos y recursos para continuar formándome y aprendiendo a la par que guiándome por el camino hacia un futuro en el que la Inteligencia Artificial esté presente en todo. Gracias por tenerme siempre en cuenta.

A Tres, porque sin él la realización de este proyecto habría sido imposible. No hay manera de terminar una investigación sin una bola de pelo ronroneando cerca. Gracias por estimular mi pensamiento.

A Natalia, por su inestimable ayuda durante estos años de carrera, por su insaciable apetito por saber más, hacer más y tener más. Gracias por impulsarme a dar lo mejor de mí mismo siempre que tengo la ocasión. Gracias por dejarme creer en mí.

A mis compañeros de grado, tanto a los que he tenido el placer de conocer mucho como a los que no. Vuestra presencia en la carrera me ha hecho salir adelante como lo he hecho, seáis conscientes de ello o no. Gracias por enseñarme y aprender conmigo.

A mi tutor, Ángel, por permitirme exprimir mis conocimientos y plantearme un reto que sólo yo podía resolver de la manera en que lo hice. Gracias por darme la oportunidad.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS

Capítulo 1: Introducción

1.1 Motivación

Hoy en día toda la información que producimos las personas tiene algún tipo de relevancia en la sociedad. Desde las transacciones financieras hasta los comentarios en las redes sociales, todo queda almacenado para ser estudiado y servir de combustible para la máquina que es el conocimiento. Como tal, las obras literarias son otra forma más de información que producimos constantemente, pero es muy importante que dicha información no se desaproveche. Para sacar partido de esto, se busca crear algoritmos y sistemas que sean capaces de reconocer patrones, clasificar cada ejemplo de un problema y, en última instancia, aprender de todo ello para ofrecer un mejor servicio.

El origen de este estudio surge de la necesidad de un sistema clasificador para un proyecto de recomendación de concursos literarios a los autores de las obras clasificadas, sirviendo como comprobación de que el autor realmente ha provisto al sistema de una obra con el género con el que el autor la define, o para clasificar libros que no han sido categorizados previamente. Esto ayudará a los autores a encontrar los concursos que se ajusten a su estilo de escritura, y protegerá al sistema de libros que no quedan comprendidos en la clasificación previamente otorgada.

Con afán de conseguir un buen clasificador, se consideraron varios sistemas de Machine Learning conocidos, ofrecidos por grandes empresas tecnológicas, accesibles al público. Las posibilidades en cuanto a complejidad y extensibilidad de las redes desarrolladas por estas empresas resultaron ser más prometedoras que una implementación “casera” de una red neuronal, por lo que este estudio comparativo arrojará luz sobre las capacidades de estos sistemas para resolver el problema propuesto.

1.2 Objetivos generales

Este estudio comparativo tiene como principal objetivo comprender y comprobar que existe un sistema de clasificación entre los principales proveedores de Machine Learning en la nube capaz de discernir el género literario de un libro u obra literaria con una precisión aceptable.

Al ser un estudio comparativo, otro de los objetivos es la obtención de una red neuronal relativamente sencilla desarrollada localmente, para establecer una precisión base a la que referirnos a la hora de contrastar con otros sistemas.

Cabe destacar que el objetivo de este estudio se centra en un clasificador de obras literarias, pero por la gran diversidad de géneros y estilos de escritura, se decidió trabajar únicamente con los libros pertenecientes a la prosa, la poesía queda excluida del experimento.

1.3 Estructura del documento

El presente documento está dividido en capítulos, cada uno tratando un aspecto del proyecto de investigación llevado a cabo.

El primer capítulo cubre la motivación del proyecto, sus objetivos generales y la estructura del documento.

El segundo capítulo trata el estado del arte referente a la tecnología utilizada para el estudio, así como brevemente las bases teóricas en las que se apoya cada recurso.

El tercer capítulo indica el alcance del proyecto junto al análisis del estudio, incluye la obtención y preprocesado de los datos utilizados en las tareas de clasificación, detalla el diseño e implementación de la red neuronal local y los requisitos a cumplir para cada sistema de clasificación alternativo. Además, se incluyen los requisitos del marco regulador con respecto a los estándares de programación en el lenguaje escogido.

El cuarto capítulo desarrolla la realización de las pruebas y estudio y comparación de los resultados obtenidos en las mismas.

El quinto capítulo trata el impacto socioeconómico del desarrollo del proyecto, detallando las posibles aplicaciones del mismo y el coste del desarrollo final.

El sexto capítulo presenta las conclusiones del estudio, incluyendo las conclusiones generales respecto de los objetivos propuestos, los problemas encontrados durante el desarrollo del mismo, los posibles trabajos futuros para el total cumplimiento de los objetivos propuestos y las conclusiones personales.

El séptimo y último capítulo es un resumen en inglés de la totalidad del proyecto, recogiendo los aspectos importantes del mismo a fin de demostrar las competencias en la lengua inglesa.

Capítulo 2: Estado del arte

Este capítulo trata sobre las bases teóricas que utiliza el proyecto para alcanzar los objetivos propuestos, además de explicar en profundidad la tecnología utilizada durante el desarrollo del estudio.

2.1 Aprendizaje Automático o Machine Learning

El aprendizaje automático, también conocido en inglés como Machine Learning (ML), puede ser definido de muchas maneras. Originalmente, se tomaba al aprendizaje automático como un estilo de programación que preparaba a un sistema para "inferir" la función que generaría los cómputos en base a una serie de ejemplos. Se consideraría que una máquina ha aprendido si los resultados que responde cuando es presentado un problema son exhibidos de manera que no siga una programación explícita (Vailant, 1984). El dominio del aprendizaje automático cae dentro de la Inteligencia Artificial, aunque según avanza el tiempo, el ML crece en direcciones que antes no se consideraban posibles.

Hoy en día el ML está presente en todo, desde los motores de búsqueda como Google hasta sistemas de delineación de parcelas agrarias (García-Pedrero, Gonzalo-Martín, & Lillo-Saavedra, 2017). Los problemas que puede resolver un sistema de ML pueden ser de clasificación, regresión, agrupamiento, predicción y más, resultando en un estilo de programación muy utilizado en la actualidad.

Dependiendo de los datos que se le provean al sistema de aprendizaje, un problema puede estar categorizado como supervisado, no supervisado o de aprendizaje por refuerzo. Existen también problemas categorizados como semi-supervisados, de transducción y de inferencia del modelo de aprendizaje (Ayodele, 2010), pero detallaremos solamente los 3 primeros. Las diferencias entre ellos son:

- Un problema de aprendizaje supervisado tiene como entrada datos etiquetados previamente, ya sea de manera manual o automática. De esta manera, el sistema sabe en todo momento si los cálculos que realiza sobre estos datos son correctos o debe corregir su función para mejorar los resultados.
- Un problema de aprendizaje no supervisado cuenta con datos no etiquetados previamente, por lo que el sistema debe trazar relaciones entre cada ejemplo dado para obtener resultados aceptables. Estos problemas suelen requerir que el sistema agrupe los datos en base a distintos factores, dependiendo del tipo de problema.
- Un problema de aprendizaje por refuerzo se basa en una premisa muy sencilla: si el sistema es capaz de generar un cómputo con un resultado favorable para la resolución del problema, éste será recompensado, mientras que, si el resultado es contraproducente para alcanzar la solución final, el sistema puede no ser recompensado o incluso puede optarse por dar un refuerzo negativo. Este tipo de problemas son comunes en tareas de búsqueda o de problemas en tiempo real, como por ejemplo la corrección de la trayectoria de un vehículo.

Para resolver este tipo de problemas existen varios modelos de aprendizaje. En este estudio nos centramos solamente en la utilización de las Redes de Neuronas Artificiales, concretamente en un ejemplo particularmente sencillo del as mismas.

Como el propio nombre indica, estos sistemas deben aprender de los datos que reciben como entrada para poder generar una función de salida. El proceso de aprendizaje varía de un modelo a otro, pero la estructura básica siempre es la misma: obtención de datos, preprocesado de datos, definición de los parámetros de aprendizaje (generación de un modelo), ejecución del bucle de aprendizaje en sí y validación y evaluación del modelo.

Los datos deben ser ajustados para que el modelo sea capaz de utilizarlos, de nada sirve darle datos puros a un sistema si no está preparado para trabajar con ellos. Las técnicas de tratamiento de datos serán analizadas más en detalle en el apartado 2.5 en este mismo capítulo.

Para cada modelo de aprendizaje automático los parámetros de aprendizaje son distintos, pero la razón de aprendizaje es un valor entre 0 y 1 que determina cuánto variará la función de salida entre iteraciones del bucle de aprendizaje. Conviene mantenerlo en valores bajos y aumentar el número de iteraciones, aunque en función de este valor en conjunto con las iteraciones del bucle se puede dar el problema conocido como mínimo local, que impide que la función de aprendizaje alcance un valor óptimo. Además de este problema puede darse el sobre aprendizaje, que se produce cuando la función aproxima con demasiada exactitud los datos de entrada, siendo capaz de obtener resultados muy positivos con estas instancias, pero perdiendo capacidad de generalización para otros ejemplos.

Una vez el modelo ha sido entrenado está listo para ser validado. Se recomienda entrenar los datos con variedad y un número similar de ejemplos de cada clase o grupo para impedir que el modelo obtenga muchos falsos positivos o negativos y no sea capaz de ajustar correctamente los valores de validación. La validación puede realizarse de varias maneras, utilizando un fragmento del set de datos de entrenamiento o un set totalmente nuevo, iterando por partes del set con validación cruzada o simplemente aplicando la función de salida con los ejemplos de validación. De la validación se obtiene una medida de precisión en función de los aciertos que ha obtenido el modelo, y se puede extraer si es un modelo válido para utilizar fuera de un entorno de desarrollo.

2.2 Redes de Neuronas Artificiales

Las redes de neuronas artificiales (RNA) son un tipo de modelo de aprendizaje automático, caracterizado por su inspiración en las estructuras biológicas a las que hacen referencia. Una red de neuronas artificial está compuesta por, como su propio nombre indica, neuronas conectadas entre sí, y utiliza estas conexiones para aprender una función de salida a partir de los ejemplos que se le da como entrada (Stergiou & Siganos).

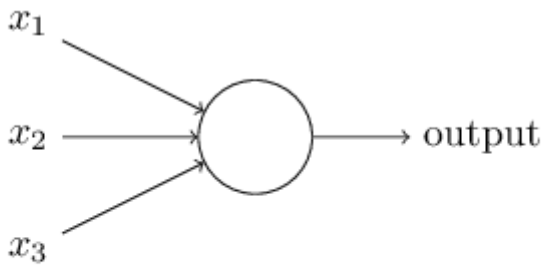
Al igual que cualquier otro modelo de aprendizaje automático, las RNA son capaces de resolver diversos problemas dentro de los descritos en el apartado anterior. Notablemente, las redes neuronales se utilizan en tareas de clasificación y regresión, es decir, aprendizaje supervisado.

Como se ha mencionado, las RNA están compuestas por neuronas, pero las conexiones que las unen poseen lo que se conoce como la función de activación, que indica si una neurona conectará con otra dependiendo de la entrada que reciba la misma. De esta manera, una red neuronal puede tener varias conexiones y funciones de activación que modificarán el comportamiento del aprendizaje a lo largo del proceso. Estas funciones de activación se definen siguiendo expresiones matemáticas dadas, cuyo objetivo es introducir no linealidad, ya que, dependiendo del problema a resolver, la función de activación juega un importante papel a la hora del aprendizaje (Vijayarekha).

A su vez, cada neurona posee una serie de pesos asociados que ajustan el grado de influencia que tiene la entrada sobre la salida hacia el resto de la red y en consecuencia sobre la función de salida. Estos pesos toman un valor aleatorio al comenzar el aprendizaje, pero se van reajustando según avanzan las iteraciones del bucle de aprendizaje para obtener resultados más próximos a la función objetivo.

Los pesos de las neuronas se van ajustando siguiendo el modelo de red implementado, que puede variar en función de cada red. Las más comunes y la implementada en este estudio son redes de estilo feed-forward, el flujo de información es similar al de una neurona biológica, pasa de una neurona a otra sin necesidad de que el resto de neuronas se activen para avanzar de capa a capa (United States of America Patent No. US5438646 A, 1995).

La red más básica en el conjunto de redes de neuronas se denomina Perceptrón, y este estudio utiliza una variación de esta red de manera que se adapte al problema en cuestión sin escalar demasiado la complejidad. Un Perceptrón Simple es una red neuronal que posee una única neurona de entrada, que recibe tantas entradas binarias como atributos tengan los datos de entrada y produce una salida binaria (Nielsen, 2017). La salida tomará un valor u otro en función de los pesos de cada entrada y del valor de la misma, además de un valor conocido como umbral, que define el punto en el que un valor obtenido a partir de las entradas producirá una salida u otra.



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Figura 1: Diagrama de Perceptrón Simple y función de salida

Esta función matemática indica el valor de la salida dependiente de si el sumatorio de 3 entradas binarias con su peso asignado supera el valor del umbral (threshold) definido. Siguiendo este modelo sencillo, el Perceptrón Multicapa implementa una versión ligeramente más compleja para acomodar diversidad de entradas y salidas.

2.3 Perceptrón Multicapa

Similar a como se ha definido en el apartado anterior al Perceptrón Simple, el Perceptrón Multicapa (Multi-Layer Perceptron, MLP) añade profundidad a este modelo de red neuronal, incluyendo varias "capas" de neuronas entre las neuronas de entrada y la neurona o neuronas de salida. La diferencia fundamental entre este Perceptrón y su análogo más sencillo es que la salida de cada neurona afecta a las neuronas de la siguiente capa, dando pie a que la función generada para la salida sea más compleja que un simple suma producto.

Este modelo de red se puede adaptar a casi cualquier tipo de problema, dado que el MLP es considerado un aproximador universal, independientemente del número de capas ocultas que posea. En el caso de este estudio, al ser un problema de clasificación, basta con ajustar la entrada para que la red se corresponda con el tamaño necesario y que la salida de la red se componga de cuantas neuronas sea necesario para representar las clases en las que se distribuyen los ejemplos.

Las capas intermedias, llamadas capas ocultas, son las capas entre la capa de entrada y la de salida. Se les denomina capas ocultas porque son las encargadas de traducir los valores de las entradas mediante funciones de transformación no lineales, buscando "simplificar" el problema para la siguiente o siguientes capas, permitiendo que un problema inicialmente complejo se reduzca a problemas resolubles con funciones más sencillas. Básicamente, una red MLP puede tratar problemas no lineales (Cybenko, 1989).

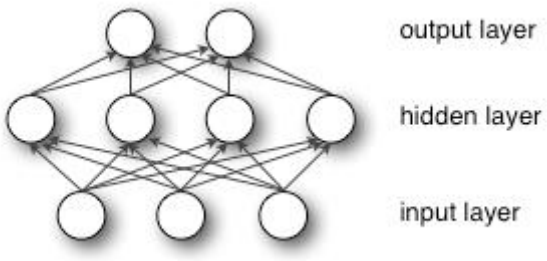


Figura 2: Diagrama de Perceptrón Multicapa

2.4 SoftMax

La función SoftMax, también conocida como función exponencial normalizada, se utiliza en el aprendizaje automático, más concretamente en las redes neuronales, para introducir un grado de probabilidad en la función de clasificación. Esta función es especialmente útil en problemas de clasificación en los que las clases son similares, resultando de gran ayuda en problemas como clasificación de textos frente a otras funciones menos apropiadas para esta tarea, como la función sigmoidea o la tangente.

$$P(y^{(i)} = k|x^{(i)};\theta) = \frac{\exp(\theta^{(k)\top} x^{(i)})}{\sum_{j=1}^{K} \exp(\theta^{(j)\top} x^{(i)})}$$

Figura 3: Ecuación de la función SoftMax

Se trata de una generalización de la regresión logística, aplicada a problemas con varias clases, se utiliza como función de coste de la clasificación penalizando los falsos positivos y negativos (UFLDL Standford, n.d.).

2.5 Descenso de Gradiente – Adam

El descenso de gradiente es uno de los algoritmos más populares de optimización, y es el más utilizado en el campo de las redes neuronales. Como tal, existen varios algoritmos que caen dentro de la familia de este optimizador.

El método que utiliza este algoritmo es minimizar la función objetivo actualizando los parámetros de manera que se distancien del gradiente de la función en el instante en el que se calcula, buscando un mínimo local.

En particular, el algoritmo Adam es un tipo de descenso de gradiente estocástico, es decir, no depende de probabilidades o cambios de gradiente. Es especialmente eficiente al tratar problemas que utilizan grandes cantidades de datos o parámetros (Diederik P. Kingma, 2015).

2.6 Tratamiento de textos

El tratamiento de datos es uno de los pasos más importantes a la hora de preparar un entorno de desarrollo, ya que unos datos corruptos o mal organizados no permiten la correcta realización de cualquier proyecto o estudio. Existen diversas partes en las que se divide la tarea del tratamiento de datos, particularmente en cuanto a tratamiento de textos.

Los datos “puros” pueden resultar incompletos, contener ruido o incluso ser inconsistentes, por lo que conviene siempre realizar un proceso de limpieza de los mismos para reducir la probabilidad de empeorar el modelo con información innecesaria e irrelevante para la resolución de un problema. Por ello, para obtener mejores resultados, siempre conviene pasar una serie de filtros antes de utilizar los datos para cualquier tarea.

En el procesamiento de textos es fundamental extraer el ruido que puede existir por caracteres especiales que no aportan nada al significado del texto. Para ello basta con pasar un filtro que detecte los caracteres ajenos al idioma en el que se representa el texto, intentando evitar que afecte a las palabras en sí. Adicionalmente, los signos de puntuación se consideran innecesarios a la hora de separar el texto, ya que para procesarlo no añaden ningún valor esencial. Se puede considerar la existencia de números en el texto como ruido, pero en función de qué tipo de números sean puede resultar útil mantenerlos en el mismo.

Otro filtro básico es el de eliminación de las denominadas *stop words*. Este tipo de palabras pueden ser útiles cuando se intenta dividir el texto en frases, pero en cualquier otro caso resultan irrelevantes para el tratamiento del conjunto de palabras. Las palabras que habitualmente se reconocen como *stop words* son los pronombres, artículos, condicionales, demostrativos, adverbios de tiempo y lugar, etc. No existe una lista definida de este tipo de palabras, depende de la interpretación de cada problema.

Una técnica popular para la reducción de la complejidad del texto se denomina *stemming*. Esta técnica consiste en reducir las palabras derivadas con prefijos o sufijos a su raíz. La técnica de *stemming* se suele identificar con simplemente deshacerse de dichos prefijos y sufijos para conseguir la raíz de una palabra. Sin embargo, la técnica conocida como *lemmatization* aspira a realizar correctamente esta diferenciación, utilizando un vocabulario de raíces existentes en el idioma tratado (Drakos & Moore, 2009). El principal problema que puede tener esta técnica es la introducción de palabras incorrectamente escritas o extranjeras, de manera que no pertenezcan al vocabulario conocido, afectando a la eficacia de este método.

2.7 Clasificación de Textos

El problema de clasificación de textos ha sido abordado en numerosas ocasiones para diferentes fines, y en la actualidad es uno de los problemas que más se utilizan para dar ejemplo de los problemas de clasificación, sobre todo orientado a mejora de los servicios de atención al cliente. En concreto, el problema de clasificación de textos más popular en la actualidad es la clasificación de "sentimiento" de un pequeño texto (generalmente reseñas o tweets) como buenos o malos hacia un servicio o compañía.

Dada la cantidad de algoritmos de clasificación existentes y la posibilidad de adaptación de muchos algoritmos a casi cualquier problema, las maneras de resolver un problema de clasificación de este estilo son casi ilimitadas. Sin embargo, no todas estas soluciones proporcionan un resultado óptimo, por lo que a lo largo de los años se han ido perfeccionando y dedicando más recursos a investigar ciertas técnicas con el propósito de orientarlas hacia la clasificación de textos.

Las comparativas de numerosos modelos de clasificación de textos dan como principales candidatos al primer puesto como algoritmo destinado a este fin son las Máquinas de Soporte Vectorial (SVM), con resultados generalmente superiores a otros algoritmos diseñados explícitamente para este tipo de problemas (Fernández-Delgado, Cernadas, Barro, & Amorim, 2014) (Pawar & Gawande, 2012). Sin embargo, otros clasificadores han demostrado tener altas capacidades para esta tarea, como son los Bosques Aleatorios (Random Forests), las redes de neuronas artificiales o clasificadores clásicos como el Naïve Bayes. De estas extensivas comparativas abstraemos que ni son necesarios tantos clasificadores para este fin, ni todos son igual de válidos para realizar esta tarea.

Las tareas de clasificación de textos en el ámbito de estudios comparativos suelen tratar de clasificar una serie de datos de dominio público como son extractos de periódicos, reseñas de películas o eventos, tweets en la red social Twitter, etc. Estos datos son fácilmente adquiribles, sin la necesidad de la adquisición de ningún volumen de datos, aunque en ocasiones sí son necesarios programas adicionales que recaben los datos a utilizarse, notablemente en el caso de los tweets. Todos estos textos se caracterizan por su corta longitud, lo que sesga el espectro observado por los problemas de clasificación de textos, aunque no les hace tener menos validez.

2.7.1 Trabajos relacionados

La clasificación de libros como tal se ve reducida al uso de elementos como la portada, la sinopsis o incluso el título de los libros, al ser complicado recopilar una serie de libros digitalizados que permitan la clasificación de estos textos utilizando el cuerpo del mismo. Por supuesto, para este fin no se restringe el uso del texto en sí, como es el caso de la clasificación utilizando la imagen de la portada y el título del libro (Chiang, Ge, & Wu, 2015).

Existen también estudios que relacionan la clase o categoría del texto con el tema que tratan, generando una vía de estudio alternativa a las palabras contenidas en el texto y centrándose más en el contexto en el que son utilizados los términos pertenecientes a la misma temática para diferenciar cuándo se trata de una categoría u otra (Petrenz & Webber, 2010).

Cabe destacar que no sólo son problemas de clasificación aquellos que tratan sets de datos previamente etiquetados. Podemos definir como problemas de clustering o agrupamiento aquellos en los que los datos a organizar no vienen clasificados previamente. Estos problemas caen en el dominio del aprendizaje no supervisado, y a pesar de no poderse categorizar como

clasificación, se basan en la distinción de patrones que agrupen distintos textos siempre que compartan rasgos comunes. Existen trabajos cuya principal función de la categorización de estos textos sin etiquetar es completar sets de datos escasamente etiquetados, haciendo posible un futuro trabajo de aprendizaje supervisado con un set de datos completo (Nigam, McCallum, Thrun, & Mitchell, 2000).

2.8 Tecnología utilizada

En este apartado se describen de manera superficial las tecnologías utilizadas en el desarrollo del estudio. Se evita un análisis más profundo de cada tecnología dado que en el caso del estudio figuran como un medio y no como la parte principal del mismo. Sin embargo, en los apartados 3.5.2 y 3.5.3 se detallan los aspectos fundamentales utilizados en el estudio de cada una de las tecnologías.

2.8.1 Python

Python se trata de un lenguaje muy popular en el desarrollo de aplicaciones de aprendizaje automático, dada su extensa variedad de librerías destinadas a este fin y su sencilla sintaxis. Este lenguaje, que no requiere de compilación, es considerado uno de los mejores para realizar tareas de investigación o experimentación, debido a la facilidad para introducir cambios en los programas sin necesidad de invertir demasiado tiempo en nuevas compilaciones.

Figura 4: Logo del lenguaje Python

Su principal punto fuerte es la inclusión de librerías como Numpy o TensorFlow que proveen las herramientas necesarias para generar modelos de aprendizaje automático y redes neuronales sin necesidad de aprender nuevas estructuras. Además, siendo un lenguaje tan común, es relativamente fácil encontrar tutoriales, ayuda o documentación sobre el uso de todas sus librerías o funciones básicas, agilizando el proceso de implementación de cualquier elemento.

El tratamiento de datos también es una de las capacidades de este lenguaje, pudiendo editar el contenido de ficheros de texto de la base de datos en pocas líneas, realizando tareas relativamente complejas en un tiempo más que aceptable.

2.8.2 TensorFlow

TensorFlow es una librería que surge a partir del grupo de Machine Learning de Google, pensada para el desarrollo e investigación de aprendizaje automático y Deep Learning. Su principal característica es que utiliza el flujo de datos con redes, en las que cada nodo representa operaciones matemáticas y las uniones entre nodos representan los arrays conocidos como tensores. El cómputo de estos modelos puede realizarse en CPUs o GPUs, permitiendo por su diseño flexible el uso de varios dispositivos para este fin.

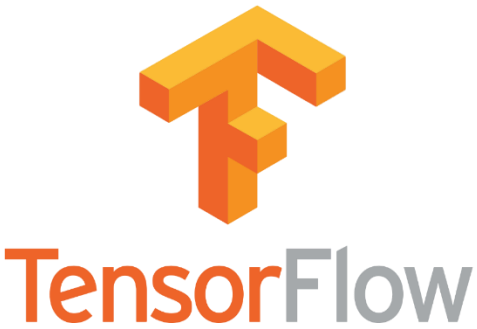


Figura 5: Logo de la librería TensorFlow

Esta librería es de bajo nivel, por lo que la definición de la estructura de la red es tan específica como se desee. Existen librerías complementarias como TFLearn que permiten un uso de TensorFlow a un nivel más alto, tratando con redes prediseñadas de los algoritmos de aprendizaje automático más conocidos. Sin embargo, editar el diseño de estas redes sigue siendo posible utilizando TensorFlow, ya que, hasta el momento de la ejecución de la red, esta es susceptible de cambiar si el usuario así lo considera necesario.

El único inconveniente que presentan estas librerías es que se encuentran en una fase muy temprana, consideradas como librerías en estadio beta. Por ello, muchas de las funcionalidades contenidas en paquetes de la librería como “learn” son inestables y susceptibles de cambio. Del mismo modo, la compatibilidad de versiones de TFLearn u otras librerías de alto nivel con TensorFlow dependen de la versión del primero, llegando a resultar inutilizables una vez se actualiza la librería base a su versión más reciente.

2.8.3 Amazon Web Services

Los servicios en red de Amazon tienen una gran variedad de sistemas que ofrecen a sus usuarios herramientas de desarrollo o análisis de datos muy útiles para empresas o particulares, aunque la orientación de la mayor parte de sus productos sea hacia grandes compañías con volúmenes de datos que aún no aprovechan su potencial.

Figura 6: Logo del producto Amazon Web Services

Entre sus servicios se puede destacar su plataforma de computación, que permite utilizar servidores, ordenadores virtuales o recursos de computación en remoto, para que los usuarios puedan aprovechar de la potencia de cómputo de una empresa como Amazon. Además, como con la mayoría de sus servicios, está integrado con otros de los productos de la empresa, pudiendo utilizar los datos almacenados en su propio servicio de almacenamiento, simplificando la tarea de introducción de datos en el sistema.

Por otra parte, su rango de servicios de análisis de datos e inteligencia artificial provee de herramientas de extracción de estadísticas de los datos introducidos al sistema, así como modelos de clasificación de imágenes, texto o voz, para ofrecer servicios de reconocimiento facial o de elementos en fotografías, el desarrollo de bots de chat mediante la IA y la transformación de texto escrito en lo que se conoce como “lenguaje natural”, intentando aproximarse al lenguaje utilizado por humanos.

Dentro del módulo de IA, Amazon provee un servicio de Machine Learning que permite crear un modelo de clasificación o regresión en base a unos datos provistos al sistema mediante su servicio de almacenamiento en línea. Mediante la modificación de parámetros y preferencias el sistema es capaz de proveer a los usuarios con estadísticas y predicciones acerca de sus datos, ofreciendo soluciones de Machine Learning para virtualmente cualquier tipo de datos.

2.8.4 Microsoft Azure Machine Learning

La plataforma de Microsoft para Machine Learning es uno de los muchos productos que ofrece esta compañía para desarrollo de soluciones web utilizando su estudio online, diseñado para uso desde navegadores web. El módulo de aprendizaje automático contiene muchas opciones, dividiendo cada apartado en los siguientes: proyectos, experimentos, bases de datos, webapps, cuadernos y modelos entrenados.

Figura 7: Logo del producto Microsoft Azure Machine Learning

Los proyectos pueden contener recursos de todas las demás categorías, sirviendo de organización entre los distintos proyectos que pueden llevarse a cabo.

Cada experimento es único y contiene las bases de datos, instancias de modelos de aprendizaje automático, métodos de tratamiento de datos y evaluación que desee el usuario. No existe un límite impuesto por Microsoft, aunque puede darse el caso de que una base de datos sea demasiado extensa para que un experimento la procese correctamente. El espacio de cómputo que se le permite a un experimento, sin embargo, depende del tipo de suscripción de la que disfrute el usuario. En la versión gratuita un usuario dispone de 10 GB para almacenar los resultados de sus cálculos, así como de espacio intermedio para los ficheros temporales que se generan en cada experimento. Microsoft ofrece una alternativa, gratuita para estudiantes y empresas, que se deshace del límite de espacio, y concede un crédito al usuario para el uso de sus herramientas de forma ilimitada. Una vez finalizado este crédito, si se desea continuar con el uso de las herramientas del entorno web se deberá abonar el importe necesario para cada producto.

El sistema de almacenamiento también va ligado a la suscripción que tenga el usuario, por lo que el límite de 10 GB gratuitos limita el tamaño y número de sets de datos que almacenar en el servicio. Utilizando su servicio estándar es posible ampliar este espacio, pero presenta los requisitos descritos en el anterior párrafo.

Los cuadernos sirven para representar los datos extraídos de una base de datos o definición de modelos de aprendizaje en los lenguajes Python 2, Python 3 o R. Su utilidad no se limita a esto, también pueden realizarse scripts de tratamiento de datos adicionales a los propuestos por la propia plataforma.

Los modelos entrenados surgen de los experimentos que realizan los usuarios, una vez entrenado un modelo es posible almacenarlo para realizar evaluaciones o predicciones utilizándolo como parte de un sistema mayor.

Finalmente, las webapps representan el estadio final de este servicio, ya que proveen al usuario de herramientas de creación de productos utilizando los modelos y datos generados la plataforma. La función de estas webapps depende enteramente de las necesidades del usuario.

Capítulo 3: Análisis y desarrollo

Este capítulo detalla el proceso de análisis del proyecto, el diseño del modelo implementado, la implementación del mismo y la estructura y modelado de los distintos servicios en la nube utilizados.

3.1 Análisis

El análisis del proyecto incluye el estudio del alcance del trabajo, los problemas que resolverá, los sistemas de los que hará uso y las alternativas de desarrollo a las escogidas para el proyecto.

3.1.1 Alcance del trabajo

Este proyecto está planteado de manera que mediante la comparación de varios sistemas o modelos de aprendizaje automático basados en redes neuronales se pueda obtener un clasificador con suficiente precisión como para resolver el problema de clasificación descrito en los objetivos generales. Para ello, el proyecto estudiará el comportamiento de las redes diseñadas más adelante, incluyendo la red implementada localmente y los servicios web de aprendizaje automático utilizados, destacando los estudiados como posibles candidatos y los finalmente utilizados en el proyecto.

El proyecto sólo tiene en cuenta los resultados obtenidos con el uso de redes de neuronas artificiales, descartando los otros sistemas clasificadores del estudio comparativo. Esta decisión se toma a fin de concretar el objetivo del estudio a un campo determinado, sin incluir una mayor amplitud de modelos de clasificación conocidos.

3.1.2 Alternativas de desarrollo

Además de los sistemas empleados para el desarrollo de este estudio, existen varias alternativas que son mencionadas en las fases de diseño y estructura de los servicios en la nube utilizados. Más notablemente se incluyen las APIs de desarrollo de modelos de aprendizaje automático ajenas a la utilizada en el proyecto.

Como se indica en detalle en el capítulo 2, la librería utilizada es TensorFlow de Google. Las alternativas encontradas para este recurso fueron la toolbox de aprendizaje automático de Matlab, muy similar a la librería de TensorFlow, pero al escoger la librería más popular y ambas ser de bajo nivel, no presentaba mejoras sustanciales respecto a la utilizada. BigML es una API diseñada para realizar aprendizaje automático, que utiliza los recursos en la nube de la compañía, de manera muy similar a cómo Google Cloud Machine Learning opera.

Respecto de los servicios web de ML, no se encontraron alternativas a los dos utilizados en el desarrollo del estudio, al menos no que se alineasen con las necesidades del proyecto. Todos los detalles acerca de los servicios web estudiados se encuentran en el apartado 3.5.

3.2 Diseño del modelo implementado

3.2.1 Marco Regulador: Especificación de estándares y normas de diseño

Para la correcta implementación y funcionamiento del modelo diseñado en el apartado 3.2.2 se siguieron los siguientes estándares y normas de manera que el entorno de desarrollo cuente con todos los recursos necesarios para la realización del proyecto:

1. El sistema operativo donde se desarrolla el programa debe ser una distribución de Linux, preferentemente de la familia Ubuntu. Cualquier versión que soporte Python 2.7 es necesaria, pero se recomienda utilizar Ubuntu 16.04 o 17.04.
2. El lenguaje de implementación debe ser Python 2.7.
3. La versión de la librería TensorFlow debe ser superior a 1.1.X.
4. No es necesario un IDE para el desarrollo del modelo, y tampoco se restringe el uso de cualquier editor de texto, siempre que sea compatible con las restricciones de tabulación de Python.
5. Se requiere un espacio de almacenamiento mínimo de 17 GB.

El lenguaje Python permite la implementación de programas sin la necesidad de un carácter delimitador de líneas del código. Sin embargo, existe un estricto sistema de indentación que debe ser respetado para el correcto funcionamiento de los programas. Este estándar dictamina que cada bucle o bloque condicional ha de ir tabulado un nivel por debajo de la línea de la condición, y sólo deberá reducirse esta indentación al regresar al bloque externo.

La definición de variables de tipo *String* puede ser realizado con valores rodeados de entrecomillado simple o doble, no existiendo diferencia entre ambos.

Para realizar comentarios en el texto se utiliza el carácter # para iniciar una línea de comentario o un comentario en una línea ya comenzada. El triple entrecomillado doble “”” sirve para realizar comentarios de documentación, ya que actúa como bloque de comentario de más de una línea.

El nombre de las variables en Python puede contener caracteres en el alfabeto y guiones bajos (_), pero no contendrá números u otros caracteres especiales. Se deben evitar los nombres ‘l’, ‘I’ y ‘O’. Para las clases es conveniente utilizar la convención de NombresConMayúscula.

Para mayor profundidad en la explicación de los conceptos de estándares marcados por Python, referirse a PEP8.

3.2.2 Definición de la arquitectura del sistema

Para el Perceptrón Multicapa utilizado como red neuronal implementada se buscaron distintas arquitecturas en Internet que pudiesen suplir las necesidades del estudio sin requerir demasiado tiempo, por el principal motivo de que esta parte sirve simplemente como un ejemplo sencillo de comparación con los demás sistemas, y bajo ningún concepto debía ser una carga de trabajo que superase la que recibirían las pruebas y el análisis de los resultados.

Se decidió utilizar la librería de Python más conocida para este fin, TensorFlow, desarrollada por Google y disponible también en otros lenguajes, pero al querer eficiencia y sobre todo código sencillo, se escogió Python.

El diseño de esta red neuronal se obtuvo de un sencillo tutorial de TensorFlow (dmesquita, 2017) en el que el autor describe el proceso de entrenamiento y pruebas de un Perceptrón para la clasificación de textos, en este caso de artículos periodísticos. Al ser una red pensada para una tarea similar a la que busca este estudio, se consideró este modelo como un buen candidato para una clasificación básica que sirviese de medida para valorar las demás.

Las únicas modificaciones realizadas al diseño son los parámetros de aprendizaje y las dimensiones de cada neurona para poder utilizar nuestro set de datos.

La red consiste de una capa de entrada con tantas neuronas como palabras existen en el vocabulario, dos capas ocultas con 100 neuronas cada una, totalmente conectadas entre sí, y una capa de salida con 20 neuronas, tantas como clases haya para clasificar. La entrada de la red es un array en el que cada posición o neurona representa una palabra, de manera que el valor de cada posición es la frecuencia de esas palabras en el texto, con valores enteros entre 0 y el tamaño máximo del texto. La salida produce un array en el que cada posición representa una clase entre las 20 posibles.

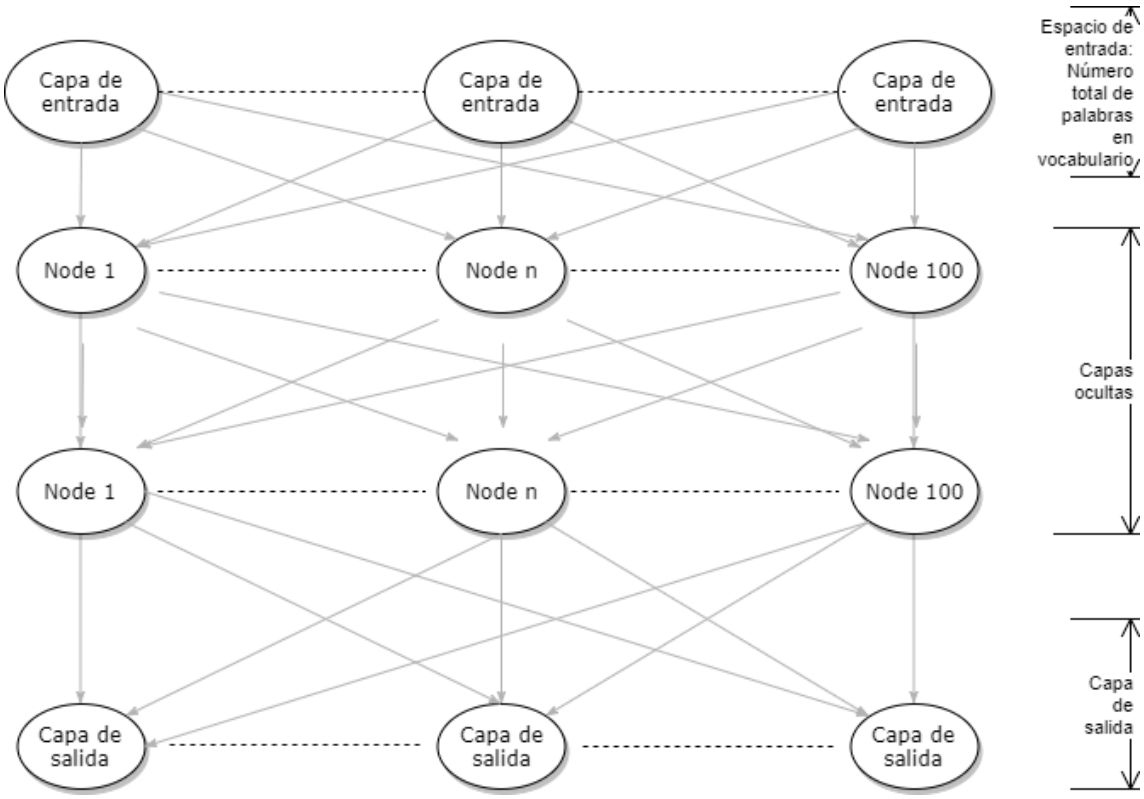


Figura 8: Diagrama del Perceptrón Multicapa

La función de pérdida que utiliza el modelo es la función SoftMax descrita en el Capítulo 2, y la función de optimización es la que emplea el algoritmo Adam, también descrito en el Capítulo 2.

3.3 Tratamiento de los datos

3.3.1 Obtención de los datos

Los datos obtenidos para el desarrollo del proyecto provienen de una extensa librería de documentos de formato .epub, de carácter personal, con un total de 10,193 libros. Dado que la

base de datos está pensada para ser gestionada con el programa Calibre, software diseñado para la gestión de eBooks y eReaders, cada libro está etiquetado con una o más categorías, además de incluir una breve descripción del libro.

La base de datos fue recopilada manualmente, por lo que la validez de los datos queda demostrada al provenir de las obras originales a las que hacen referencia los títulos.

3.3.2 Características extraídas

Para poder entrenar la red neuronal que se implementó localmente, y potencialmente las demás, se barajaron distintas ontologías relacionadas con qué definía un género literario, para intentar aproximar los atributos presentados a la red de manera que el aprendizaje fuese más sencillo. Inicialmente se contaban entre estas características aquellas como el número total de palabras del libro, la frecuencia de los *n* términos más frecuentes, junto a dichos términos, la longitud media de las palabras y la exclusión de *stop words* o palabras "ajenas" al cuerpo principal del documento (aquellas como artículos, preposiciones, interjecciones, etc.).

Sin embargo, tras una larga serie de pruebas y errores, se decidió simplemente dar como entrada el texto en sí, no sin sanearlo previamente para evitar la presencia de caracteres ajenos al propio texto. La razón para descartar la extracción de características es principalmente la de que la red neuronal implementada tan sólo requería de las palabras presentes en cada texto. Para evitar modificaciones innecesarias al modelo se utilizó el formato propuesto por el autor original. Siguiendo este formato, se necesitaba una recopilación de las palabras existentes en todos los textos, para poder entrenar el modelo. De este modo, se realizó un vocabulario con las palabras presentes en los 10,193 textos, ordenándolas por frecuencia de aparición.

3.3.3 Modificaciones realizadas

La primera modificación necesaria fue el cambio de extensión de los ficheros .epub, característicos de los libros electrónicos. Para ello se empleó la herramienta de conversión que proporciona el programa de gestión Calibre, que convierte el formato del fichero de entrada a una gran variedad de formatos de salida, entre ellos el fichero de texto convencional que, al ser sencillo y fácil de manipular, se eligió como formato para operar con los libros. La portada y cualquier imagen asociada se elimina del texto y lo que obtenemos es una serie de ficheros con extensión .txt que contienen el texto puro de cada página, manteniendo las tabulaciones, los saltos de línea y demás características. El único problema que tal vez pudo afectar a la "pureza" de los datos es que no se puede establecer un rango de páginas a transcribir, es decir, excluir la información de publicación del libro o los agradecimientos e introducciones/prólogos. Por tanto, todos los textos contienen también estas partes del libro dentro de los datos utilizados en la clasificación.

El texto no fue utilizado como entrada sin modificar, a pesar de no extraer características sí que se eliminaron ciertas partes del texto con el fin de mejorar la clasificación, de acuerdo a la información obtenida durante la investigación sobre clasificación literaria. Como indicaban la mayoría de artículos referentes a este tema, para una correcta clasificación de cualquier texto se deben evitar los elementos que sean irrelevantes para la clase en sí. Como tal, un género literario no viene definido por el número de párrafos o la longitud de las oraciones, a menos claro que estemos clasificando poesía, y al no ser el caso de este estudio, nos permitimos eliminar todo tipo de separación entre el texto que no fuese un espacio. Por lo tanto, tabulaciones, saltos de página, signos de puntuación y caracteres especiales como dobles comillas o paréntesis fueron eliminados.

A través de las diferentes pruebas realizadas antes de obtener el modelo final de clasificación se barajaron la eliminación de más elementos como interjecciones, artículos o preposiciones, incluso palabras menores de una longitud definida. La única modificación realizada finalmente en los datos proporcionados como entrada fue la reducción a la raíz de cada palabra, técnica conocida como *stemming*. Para ello se utiliza una librería que contiene un extenso vocabulario en el idioma deseado para llevar a cabo este proceso (Porter). Por desgracia, los textos contenían palabras escritas incorrectamente, además de tecnicismos y palabras extranjeras que la librería no tenía capacidad de procesar. Por todo ello se decidió revertir los cambios al uso de palabras completas.

Adicionalmente, las numerosas pruebas nos permitieron ver que los datos estaban extremadamente sesgados, la población de varias clases superaba con creces las de muchas otras, teniendo un total de 70 clases distintas con representaciones variando entre 1530 la clase con más instancias y 1 única instancia las clases menos representativas. Dada esta abismal diferencia, y como las 50 clases con menor representación no superaban las 100 instancias, decidimos eliminarlas del conjunto de datos. Los datos siguen estando mal balanceados, siendo el nuevo mínimo de 100 instancias, pero se consideró un paso en la dirección correcta a la hora de entrenar mejor los modelos. De este modo, el número de textos se redujo de 10,193 a 9,347.

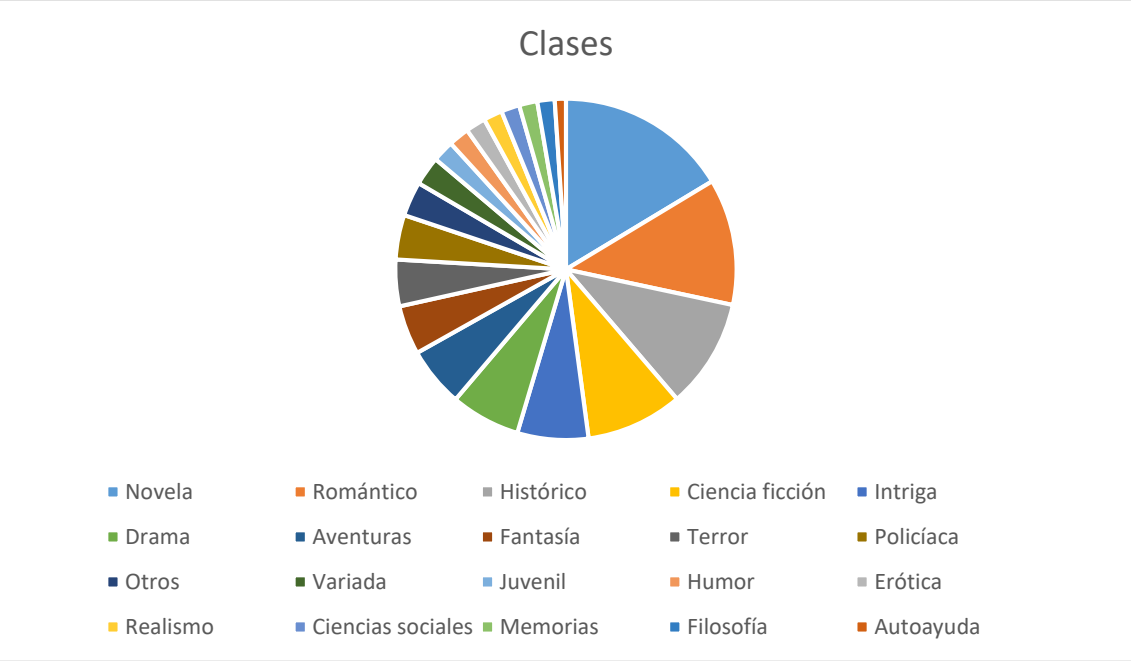


Gráfico 1: Distribución de clases en la base de datos de 9,347 libros

Aunque en principio las instancias ya estaban aleatoriamente distribuidas (al estar los textos organizados por autor, no por clase) se introdujo una nueva ordenación aleatoria para obtener una distribución lo más homogénea posible entre los ejemplos.

Con todas las modificaciones descritas anteriormente, los datos que utilizarían los modelos para entrenar y validar se dividieron en dos segmentos con proporción 70-30, consistiendo de un fichero con tantos ejemplos como textos, con cada línea estando compuesta por el texto completo, ya tratado, y la clase correspondiente al mismo.

Para los sistemas de clasificación en la nube, utilizar esta configuración de datos resultó ser imposible, ya que los recursos que permitían los servicios web no alcanzaban a procesar la

enorme cantidad de datos que se les daba como entrada (4.5 GB de datos). Con ánimo de permitir procesar los datos a estos servicios, se volvió a analizar la posibilidad de utilizar la extracción de características mencionada en el apartado anterior. Se decidió utilizar simplemente los *n* términos más frecuentes, eligiendo 100 como el número en cuestión. El nuevo set de datos contendría las 100 palabras más frecuentes en cada texto, ordenados por frecuencia, y la clase a la que pertenece dicho ejemplo. Otros atributos como el título o el número total de palabras fueron barajados, pero se consideró irrelevante para los sistemas utilizados.

Finalmente, se generó otro set de datos alternativo, para realizar pruebas comparativas, reduciendo el set inicial de 10,193 libros a uno de 2,000. La reducción viene dada a causa del balanceo total de los datos, consistiendo este nuevo set de 100 instancias de cada una de las 20 clases escogidas. Con los datos mejor balanceados, aunque escasos, se pretendían obtener resultados más consistentes.

3.4 Implementación del sistema de clasificación

La implementación del diseño se realizó en Python 2.7, utilizando la versión 1.3.0 de la librería TensorFlow.

3.4.1 Ajustes de la implementación

El preprocesado de los datos se modificó para acomodar el uso de nuestro set de datos, y la creación del vocabulario se realizó previamente para evitar carga computacional innecesaria a la hora del aprendizaje. Al probar el Perceptrón con varios sets de datos a medida que se realizaron las pruebas, la estructura del preprocesado de datos varió en varias ocasiones, pero no de manera destacable.

En un comienzo, la implementación no se diferenciaba demasiado del modelo original, incluyendo entrenamiento y pruebas del modelo en el mismo fichero. Pero después de varias pruebas resultantes en una caída del sistema debido a la enorme carga computacional, se decidió guardar el modelo en un fichero externo una vez terminado el entrenamiento para la posterior realización de las pruebas. Este fichero externo retenía los datos de los tensores que utiliza la librería, lo que requiere del nuevo programa de pruebas que inicialice los mismos tensores para recuperar los datos desde el fichero externo. De este modo se consiguió repartir las tareas de manera que a pesar de que en las pruebas hubiese algún problema, el modelo entrenado no sufriría pérdidas.

3.4.2 Entrenamiento del modelo

Para poder probar el modelo, es necesario entrenarlo primero. Para ello, se hicieron muchas pruebas de volúmenes de datos que el ordenador que facilitaba el entorno de desarrollo pudiese soportar. Con el diseño implementado se entrenaba mediante paquetes reducidos de ejemplos a partir del set total, buscando evitar el colapso de la red al intentar dar como entrada todos los ejemplos simultáneamente, resultando en un entrenamiento localizado en cada paquete por tantas iteraciones como fuesen definidas.

Se comenzaron con entrenamientos sencillos de pocos textos, buscando que el modelo demostrase que funcionaba. Una vez confirmado que el código era correcto y que nuestros datos se amoldaban bien al modelo, se realizaron los entrenamientos correspondientes a los sets de datos extraídos.

Utilizando los sets de datos con los textos completos, el proceso de entrenamiento resultó ser extremadamente lento, de manera que 100 iteraciones por los datos alcanzaban fácilmente las 5 horas de ejecución. Esto hizo que se planteasen los cambios mencionados respecto a los sets de datos, ya que la utilización de los textos completos requería de demasiado tiempo de cómputo, aunque los resultados de las pruebas determinarían si este tiempo merecería la pena.

Tras el cambio de set de datos, el entrenamiento se realizó en tiempos mucho más razonables, debido enteramente a que el set de entrada dejaba de contener cerca de 2 millones de palabras y pasaba a tener tan solo 100.

Con cada entrenamiento se generaba un modelo entrenado, y lo que los diferenciaba eran parámetros como la razón de aprendizaje, el número de iteraciones que realizaba durante el aprendizaje a lo largo de los datos y el tamaño del paquete de ejemplos que se le daba como entrada a la red.

3.5 Estructuras de los sistemas estudiados

Este apartado contiene la información recabada acerca de los distintos servicios web de Machine Learning, tanto de los que han sido usados para el estudio como de los que por motivos de incompatibilidad con el problema no pudieron ser empleados. Para cada servicio se analizan las prestaciones ofrecidas, y para aquellos que cumplen los requisitos para la ejecución de un clasificador de textos utilizando modelos de redes neuronales se desarrollan detallando las distintas modalidades de clasificador que ofrecen.

3.5.1 Google Cloud Machine Learning

El primer entorno web estudiado fue el perteneciente a la compañía Google. Al ser una empresa que requiere tanto del análisis de los datos que recaban, se pensó que su servicio de aprendizaje automático en línea sería útil para el proyecto. Sin embargo, los sistemas ya implementados que provee Google en su Cloud Machine Learning son muy limitados a los servicios que ellos mismos proveen:

- Cloud Vision: Se trata del modelo de reconocimiento de elementos en imágenes que utiliza Google. Por supuesto, este modelo no sirve para clasificar textos y mucho menos libros.
- Cloud Speech: Este modelo convierte ficheros de audio, ya sea en tiempo real o de manera asíncrona, en texto empleando el idioma indicado de entre 80 posibles. De nuevo, el modelo no se ajusta a las necesidades del proyecto.
- Google Translate: Uno de sus modelos más conocidos, permite la traducción cada vez más acertada de muchos idiomas. Una vez más, el modelo no se corresponde con un clasificador de texto.

Además de los modelos ya preparados por Google, el servicio de Cloud Computing permite a los usuarios utilizar los recursos en línea de la empresa, no de manera gratuita, para realizar los cómputos o ejecutar los programas que cada cliente desee. Entre otros de sus servicios se encuentra el módulo de Machine Learning, que dispone de una API que permite a los usuarios crear modelos de modo local en sus entornos de desarrollo para posteriormente emitir un "trabajo" que el sistema de cómputo de Google se encargará de entrenar o evaluar. Este servicio se alinea en cierta medida con los requisitos del proyecto, pero requiere que el modelo sea realizado por el usuario, como es lógico.

La estructura del servicio de Machine Learning permite a los usuarios con un modelo de aprendizaje automático realizar operaciones que posiblemente no serían tan sencillas de realizar en sus propios entornos de desarrollo, por capacidad de cómputo o por restricciones de tiempo. Para nuestro modelo ya implementado habría resultado una buena comparativa en cuanto a tiempo de cómputo, pero al realizarse del mismo modo el proceso de entrenamiento y evaluación, carecía de sentido utilizarlo para comparar los resultados del propio modelo.

A pesar de intentar hacer el modelo compatible con esta herramienta para poner a prueba la capacidad del servicio, resultó imposible por restricciones temporales, y dado que no aportaría ningún dato nuevo al estudio, se descartó el uso de este sistema.

3.5.2 Amazon Web Services

El entorno web de Amazon también provee a sus usuarios de varios servicios de computación en línea, así como de almacenamiento de datos y demás. Muy similar a la estructura del servicio de Google, Amazon también requiere que los usuarios hagan un registro completo incluyendo información bancaria para las transacciones por sus servicios. Sin embargo, pequeñas partes de las prestaciones disponibles tienen opciones para desarrolladores que permiten el uso de sus herramientas sin necesidad de un pago previo.

Entre los servicios prestados por Amazon destacamos el módulo de Machine Learning. Esta herramienta permite a los usuarios utilizar sus propias bases de datos para alimentar el modelo de aprendizaje automático provisto por Amazon, realizando el tipo de aprendizaje correspondiente a la tarea asignada. Mediante un sencillo proceso dividido en pasos, el sistema guía al usuario en la realización de un modelo a evaluar.

3.5.2.1 Preparación del modelo

Amazon posee un servicio de almacenamiento en la nube llamado S3 (Simple Storage Service). Mediante este servicio un usuario puede almacenar todo tipo de datos en su espacio en la nube. En el caso de este estudio, se utilizó para almacenar el set de datos compuesto por las 100 palabras más comunes en cada texto, utilizando tanto el fichero de 20 clases completo como el balanceado.

Tras almacenar los datos para los experimentos, es necesario indicar al sistema el tipo de entradas que tendrá que tratar el clasificador (numérico, categórico, texto o binario), así como el campo que corresponde a la clase, y de qué tipo es. El sistema también preguntará si los datos tienen un campo identificador, pero los ejemplos en nuestro set de datos no vienen identificados.

Una vez definidos los datos, el sistema provee al usuario con la elección de dejar que el sistema genere un modelo propio en base al tipo de datos que va a utilizar, pero también permite que el usuario decida el tipo de modelo que utilizará. El modelo por defecto utiliza una “receta” por defecto, que se trata de una configuración estándar que usa la red. Al querer comparar su modelo con el de otros servicios, se mantuvo esta “receta” sin modificar. Por otra parte, el sistema permite modificar también los parámetros de aprendizaje y la cantidad de datos destinada a entrenamiento y validación, de manera secuencial o aleatoria.

3.5.2.2 Entrenamiento del modelo

Cuando el usuario tiene los datos y el modelo correctamente definidos, el sistema entrena el modelo generado con los datos provistos, y en cuestión de minutos publica un resultado que el usuario puede consultar para evaluar su modelo. Estos resultados contienen una razón de

precisión entre 0 y 1, siendo 0 la base que considera Amazon que debe tener un modelo, empleando un clasificador como ZeroR para determinar esta base (ZeroR clasifica todas las instancias usando la clase predominante, por tanto, tiene una razón de acierto proporcional al número de ejemplos de esa clase en el set de datos). Junto a esta valoración de precisión, Amazon provee al usuario con una matriz de confusión que ilustra el porcentaje de ejemplos clasificados correcta e incorrectamente durante la evaluación, y da la opción de obtener la tabla completa, ya que la mostrada en el servicio web tan solo contiene una parte de las clases.

3.5.3 Microsoft Azure Machine Learning

El Microsoft Azure ML Studio permite, de similar manera que el servicio de Amazon, generar modelos de cualquier tipo de tarea de aprendizaje automático utilizando los datos que requiera el usuario, siempre que estén almacenados en su servicio de almacenamiento. Este servicio es el más diverso en cuanto a opciones de desarrollo, permite que los usuarios diseñen todo el procedimiento, desde el formato de la base de datos hasta el número de pruebas y comparaciones que realizará el sistema con tantos modelos de aprendizaje como se deseen.

El aprovechamiento de todos estos recursos conlleva un coste monetario, pero Microsoft habilita ciertas funcionalidades de manera gratuita para estudiantes, permitiendo llevar a cabo el desarrollo del estudio sin requerir un pago previo.

3.5.3.1 Preparación del modelo

La realización de cada experimento requiere de la creación de un esquema que Azure ML Studio seguirá con el objetivo de entrenar y evaluar el modelo propuesto. Este esquema consta de una base de datos, un nodo de tratamiento de esos datos, un modelo de aprendizaje automático, en nuestro caso relacionado con redes neuronales, un nodo de entrenamiento de la red con los datos seleccionados, un nodo de “puntuación” utilizando los datos de evaluación y el modelo entrenado y por último un nodo de evaluación que muestra los resultados obtenidos de manera gráfica.

Al igual que Amazon, este sistema no es capaz de procesar la base de datos que contiene la totalidad de los textos, resultando imposible hacer un modelo utilizando este set. Por ello, se utilizaron los sets correspondientes a las 100 palabras más comunes y el set de datos balanceado con 100 instancias para cada clase. Ambos sets contienen ejemplos de las 20 clases definidas en el tratamiento de datos.

Una vez subida la base de datos al sistema de almacenamiento de Microsoft, Azure ML Studio permite que se utilice este set de datos como módulo en su esquema para entrenar una red. Utilizando el nodo de tratamiento de datos, se divide la base de datos en dos fracciones con proporción 70-30 para entrenamiento y validación.

Los modelos disponibles en esta herramienta son muy variados, pero por desgracia sólo contienen 2 modelos relacionados con redes neuronales, y tan sólo uno de ellos es compatible con nuestro modelo, al ser un clasificador multiclase. Por ello, para estos experimentos se utilizó el modelo propuesto por Microsoft de una Red Neuronal de clasificación multiclase.

Una vez conectados todos los elementos descritos, se agregan el nodo de “puntuación” y el nodo de evaluación, resultando en un esquema listo para entrenar.

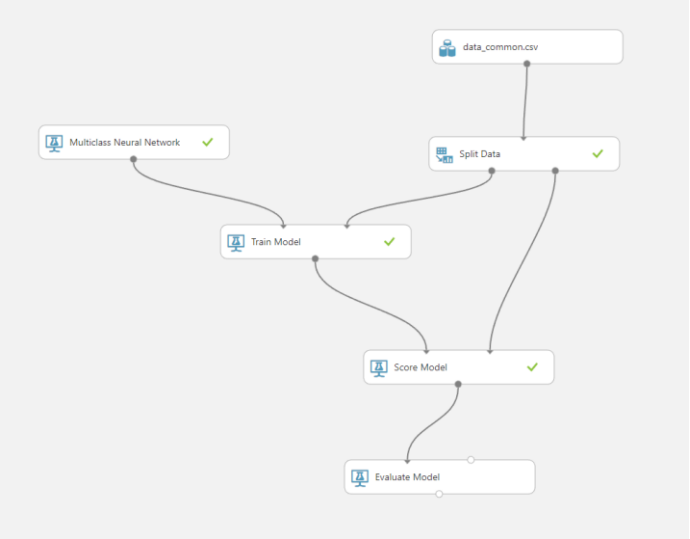


Figura 9: Esquema de entrenamiento de un modelo en MS Azure

3.5.4.2 Entrenamiento del modelo

El servicio permite modificar los parámetros de aprendizaje del modelo de aprendizaje automático, así como el número de neuronas ocultas, en el caso de nuestra red, las iteraciones de aprendizaje y los pesos de cada neurona. Permite también definir cómo estarán conectadas las neuronas entre sí, pero en nuestro caso decidimos mantener el caso totalmente conectado para todos los experimentos.

3.5.4 IBM Watson, BigML y SAP

Este último punto reúne 3 servicios de aprendizaje automático, y se decide hacer de este modo porque en un comienzo eran posibles candidatos para las pruebas realizadas en el estudio, pero tras comprobar el tipo de servicios que cada uno provee, se descartaron totalmente.

En el caso de IBM Watson, el módulo de aprendizaje automático en la nube aún no está funcionando, según la propia página web del servicio se encuentra en desarrollo en la actualidad, y no existía más que un formulario para suscribirse a su lista de correo para ser notificado una vez pongan en disposición del público su entorno web de Machine Learning.

En cuanto a BigML, se trata de una API de alto nivel para realizar modelos de aprendizaje automático, lo cual equivale a la librería TensorFlow utilizada para modelar el Perceptrón. Dada la similitud con el recurso ya utilizado, se descartó esta API por no aportar novedad o diferenciación al estudio, además de no proveer modelos ya implementados para realizar pruebas.

Por último, la empresa SAP también provee a sus clientes con modelos de aprendizaje automático que pueden utilizar con sus propias bases de datos. El problema surge cuando se inspeccionaron los modelos provistos por esta compañía, ya que todos están orientados a empresas y servicios de gestión empresarial, teniendo modelos de análisis de datos de ventas, clientes o estadísticas. Ningún modelo propuesto por este servicio se aproximaba a las necesidades del estudio, por lo que fue descartado al carecer de un entorno web de desarrollo de modelos o poseer un modelo propio de clasificación de textos.

Capítulo 4: Evaluación

En este capítulo se detallan las pruebas realizadas con los distintos modelos empleados para la clasificación de libros según su género literario. Se entienden como pruebas tanto el entrenamiento de cada red para la posterior validación de la misma, con las variaciones de parámetros estimadas relevantes para obtener datos variados, como la propia validación y los resultados de precisión obtenidos tras la evaluación de cada modelo.

4.1 Entorno de pruebas

El entorno empleado para las pruebas del modelo local del Perceptrón Multicapa es un ordenador con el sistema operativo Ubuntu 17.04, la versión de Python 2.7 instalada y la versión 1.3.0 de TensorFlow instalada.

El entorno empleado para las pruebas de los modelos en red es el sistema operativo Windows 10 Education, en el navegador Google Chrome 60.0.3112.32.

4.2 Descripción de las pruebas

Cada prueba realizada con cada modelo consta de una serie de parámetros de aprendizaje dentro del modelo escogido en cada caso. En este apartado se detallan los rangos de variación que sufren dichos parámetros para cada modelo.

4.2.1 Entrenamiento de la red

Para el Perceptrón Multicapa destacamos varios parámetros de aprendizaje que pueden variar dentro del modelo. Por mantener un diseño constante de la red, el número de capas ocultas y el número de neuronas de cada capa se mantendrá constante, a fin de no añadir infinidad de combinaciones al desarrollo de las pruebas.

Los parámetros modificados son:

- Tamaño del paquete: número de textos que se dan como entrada a la red para entrenar en cada iteración de aprendizaje.
- Número de iteraciones: total de iteraciones que se ejecutará el bucle de aprendizaje.
- Razón de aprendizaje: valor según el cual la red aprende y modifica sus funciones para aproximar más lenta o rápidamente un valor.

4.2.2 Entrenamiento de otros modelos

Para los otros modelos el entrenamiento se lleva a cabo dependiendo del servicio en cuestión.

Microsoft Azure Machine Learning Studio presenta un diseño que permite la realización de varios modelos en el mismo experimento, lo cual nos permite realizar el entrenamiento de varios modelos de la misma red utilizando distintos parámetros todo de una vez, si bien no simultáneamente debido a la suscripción estándar existente. Los parámetros que permite modificar este servicio son:

- Razón de aprendizaje: este valor puede tomar un único valor o varios valores de manera que cada prueba entrene el modelo con los valores indicados en el set, o el rango

definido entre los valores que hacen de cota superior e inferior. El propio sistema se encarga de escoger la mejor combinación y muestra los resultados de la misma.

- Número de iteraciones: al igual que la razón de aprendizaje es posible decidir si se quiere un único valor o un rango de valores para que el clasificador entrene con todos ellos.
- Diámetro inicial de los pesos: permite definir el orden en el que se verán inicializados los pesos de las neuronas (por defecto es 0.1).
- Momento: permite ajustar el valor de oscilación de la función de pérdida cuando utilizamos el descenso de gradiente. Sirve para balancear las actualizaciones que sufre la función de pérdida según la razón de aprendizaje (por defecto es 0).
- Número de nodos ocultos: el número de neuronas ocultas que existen en la red, no permite definir cada capa independientemente en un caso totalmente conectado.
- Especificación de capas ocultas: por defecto conecta todas las neuronas ocultas entre sí, por lo que requiere una definición manual en caso de querer una distribución distinta de las neuronas.
- Tipo de normalizador: en caso de poder normalizar los datos, existen varios algoritmos de normalización (por defecto es Min-Max). Nuestros datos al ser texto no serán normalizados.

Amazon Web Services permite que el usuario defina su modelo de entrenamiento desde los datos de entrada hasta los parámetros de la red neuronal. Sin embargo, Amazon no permite que se edite la estructura de la red neuronal, ni la razón de aprendizaje, utiliza unos valores predeterminados en base a los datos dados como entrada (en el caso de este estudio, analizando los logs de cada prueba se descubre que utiliza los valores de razón de aprendizaje en el rango {0.01, 0.1, 1, 10, 100}). Los parámetros editables son:

- Tamaño máximo del modelo: este parámetro pone un límite de espacio a cuánto puede llegar a pesar el modelo generado. El máximo impuesto por Amazon es de 2000 MB, aunque permite reducirlo si el objetivo es tener un modelo bajo unos requisitos de tamaño.
- Número máximo de iteraciones: el máximo número de iteraciones de aprendizaje que permite Amazon son 100, pero permite definir un número menor.
- Mezclar los datos de entrada: este campo permite aleatorizar el orden de los datos de entrada si el usuario no ha realizado este paso al generar la base de datos.
- Tipo de regularización: similar al momento en la solución de Microsoft, este campo permite elegir entre L1 y L2 para agregar una variable de estabilidad a la función de pérdida. L1 utiliza el valor absoluto de los pesos y L2 su valor cuadrático.
- Cantidad de regularización: permite establecer la cantidad de variabilidad que permitirá el parámetro de regularización. Da a escoger tres valores y adjunta una anotación a cada uno: 1e-6 – Suave, 1e-4 – Medio, 1e-2 – Agresivo.

4.2.3 Validación de los modelos

La validación de todos los modelos sigue el mismo patrón: división del set de datos al 70-30, entrenamiento con el 70% y validación con el 30%. La división se encarga de hacerla cada servicio o, en el caso de la red local, se hace programáticamente. Esta división nos permite aprender con un número razonable de ejemplos y tener una variedad aceptable de ejemplos de validación. En función del set de datos utilizado, esta afirmación se mantendrá o no, ya que el set de 9,347 libros no contiene una distribución equitativa de los datos en cuanto a representación.

Tras la validación se analizan los resultados obtenidos mediante una medida de precisión, además de una matriz de confusión que muestra las clases clasificadas erróneamente, dando indicaciones de dónde se encuentran los falsos positivos y negativos de cada validación. Esta matriz es extremadamente útil en el supuesto de que se desee mejorar el funcionamiento de la red, concentrando los esfuerzos en reducir en lo posible la ocurrencia de dichos resultados erróneos.

4.3 Resultados obtenidos

Este apartado recopila y analiza los distintos resultados obtenidos tras las pruebas realizadas, ilustrando con gráficos comparativos la precisión o el rendimiento de cada sistema con respecto a los demás.

Debido a las características de los sistemas web estudiados, los resultados obtenidos de Microsoft Azure Machine Learning Studio y Amazon Web Service reúnen los resultados en una sola estadística con el mejor modelo de los entrenados, ya que realiza una preselección antes de mostrar los valores de validación al usuario.

4.3.1 Resultados de MS Azure

Los resultados del servicio de Microsoft incluyen una medida de precisión global, una medida de precisión media y medidas de micro y macro precisión y recuperación, además de una matriz de confusión de cada clase.

Experimento 1: 100 palabras comunes, datos sin balancear, 20 clases (Cross Entropy)

En este experimento se realizó el aprendizaje con razones de aprendizaje entre 0.001 y 1, utilizando 10 valores intermedios según la herramienta de creación de rangos, y se realizaron pruebas con iteraciones de valores 10, 25, 50, 75 y 100. El mejor valor para la razón de aprendizaje es el de 0.01, y el número de iteraciones es 10. La función de pérdida escogida por MS Azure para este experimento es Entropía Cruzada.

Overall accuracy	0.348787
Average accuracy	0.934879
Micro-averaged precision (F1)	0.348787
Macro-averaged precision	NaN
Micro-averaged recall	0.348787
Macro-averaged recall	0.238024

Tabla 1: MS Experimento 1

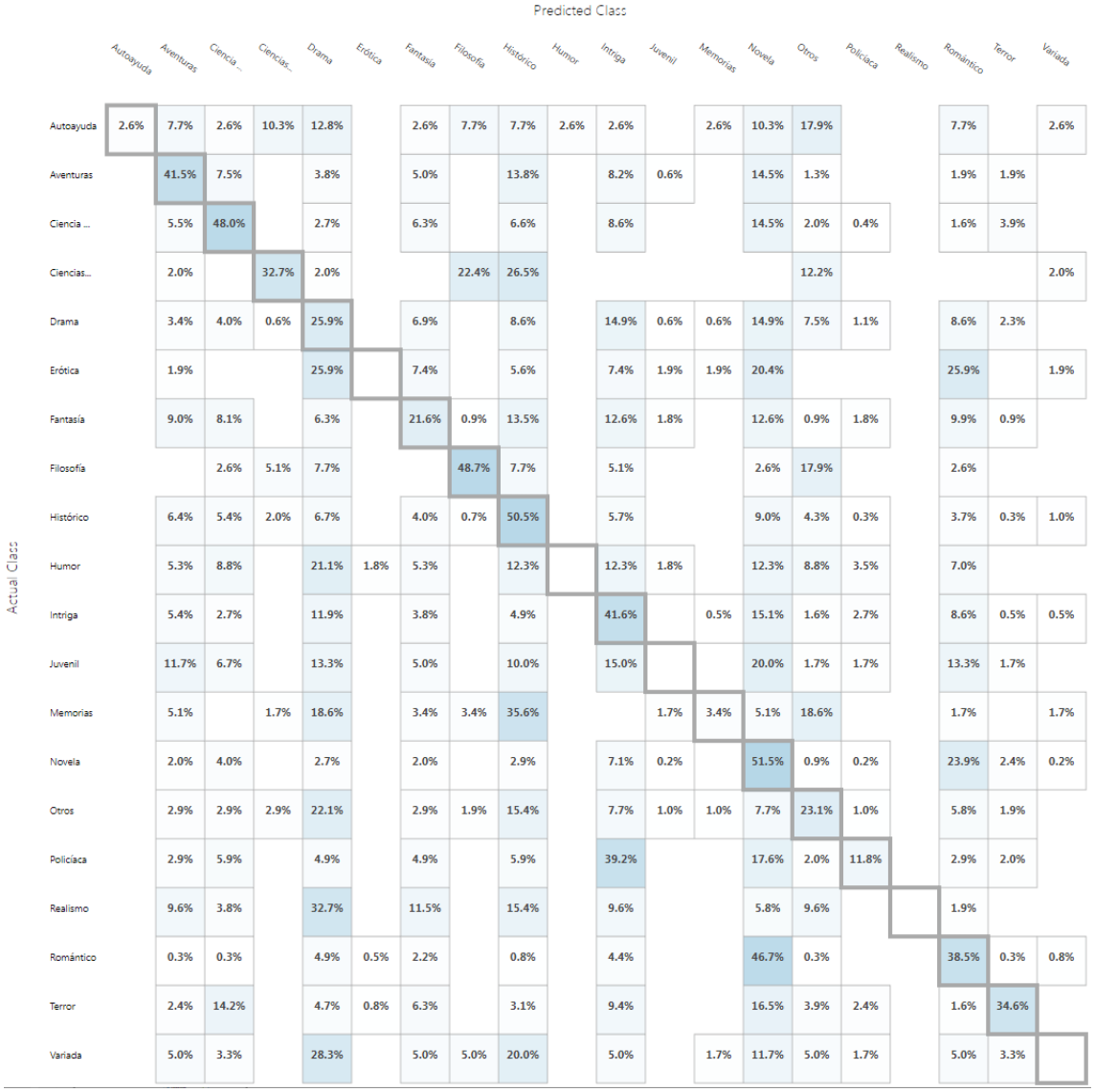


Figura 10: MS Experimento 1 Matriz de Confusión

En la matriz de confusión se aprecian en un azul más oscuro los porcentajes de clases con mayor número de predicciones, ya sean falsos negativos o aciertos. De manera genérica podemos observar que la diagonal, que representa los verdaderos positivos, contiene para casi todas las clases un valor en colores más oscuros que el resto de la fila, significando que al menos la mayor parte de las clasificaciones pertenecen a la clase correspondiente. Existen bastantes confusiones en clases como *Romántico*, cuya principal clase de confusión es *Novela*. Vemos ocurrencias similares en *Ciencias Sociales* con *Filosofía* e *Historia*, confusión perfectamente comprensible al verse las dos últimas comprendidas dentro de las ciencias sociales, por lo que el contenido de estos textos debe tener gran similitud.

Podemos observar que existen algunas clases para las que ningún ejemplo es correctamente clasificado, como *Erótica, Humor, Juvenil, Realismo* o *Variada*. Esto puede deberse a varios factores, una representación escasa en el grueso de la base de datos o una similitud extremadamente grande con otra clase (como es el caso de *Memorias*, que recibe un 3.4% de positivos verdaderos mientras que un 35.6% es clasificado como *Histórico*). Casos como el de *Erótica* son interesantes, dado que ningún ejemplo es clasificado como la clase verdadera, pero la clasificación está mayoritariamente repartida entre *Romántico, Novela* y *Drama*, clases que podrían fácilmente confundirse con esta clase.

En general extraemos que el clasificador, pese a realizar una clasificación catalogable como mediocre, es capaz de, en su error, trazar relaciones entre clases aparentemente distintas por su gran similitud estructural y de contenido.

Experimento 2: 100 palabras comunes, datos balanceados, 20 clases (Squared Error)

Este experimento modifica el anterior disminuyendo el set de datos de entrenamiento a 2000 libros, pero con un porcentaje equitativo de representación en cuanto a clases. El set de validación son la totalidad de los libros menos los 2000 libros utilizados en el entrenamiento. Los parámetros de aprendizaje son los mismos que para el Experimento 1. El mejor valor para la razón de aprendizaje es 0.2 y el número de iteraciones 100. Como función de pérdida, MS Azure define el Error Cuadrático Medio como mejor opción para maximizar el porcentaje de acierto.

Overall accuracy	0.193579
Average accuracy	0.923198
Micro-averaged precision (F1)	0.193579
Macro-averaged precision	0.183014
Micro-averaged recall	0.193579
Macro-averaged recall	0.223563

Tabla 2: MS Experimento 2

Predicted Class

Actual Class	Autoayuda	Aventuras	Ciencia ...	Ciencias...	Drama	Erótica	Fantasía	Filosofía	Histórico	Humor	Intriga	Juvenil	Memorias	Novela	Otros	Policíaca	Realismo	Romántico	Terror	Variada
Autoayuda	58.6%			13.1%	2.0%			3.0%	1.0%	2.0%	1.0%	2.0%	3.0%		4.0%	3.0%	4.0%			3.0%
Aventuras	2.6%	21.7%	8.3%	1.3%	3.8%	3.4%	5.5%	0.8%	7.2%	6.8%	3.6%	5.5%	2.6%	2.3%	2.8%	5.7%	4.7%	3.2%	3.4%	4.7%
Ciencia ...	1.4%	10.4%	21.9%	1.8%	4.9%	2.8%	5.7%	2.3%	4.1%	3.3%	6.6%	5.2%	1.6%	2.6%	3.7%	9.0%	3.7%	2.9%	2.0%	4.0%
Ciencias...	6.7%	1.2%	5.5%	22.7%	4.3%	1.8%	1.8%	10.4%	3.1%	3.1%	0.6%	7.4%	6.1%		4.9%	4.3%	6.7%	3.7%	1.2%	4.3%
Drama	2.4%	4.9%	4.2%	3.2%	11.7%	5.5%	4.9%	2.1%	7.0%	5.7%	4.9%	5.3%	4.5%	3.1%	5.5%	5.3%	6.5%	5.7%	3.1%	4.5%
Erótica	2.8%	4.5%	1.7%	1.1%	3.4%	29.4%	3.4%	0.6%	1.1%	4.5%		7.9%	7.3%	4.0%	4.0%	2.3%	9.6%	4.5%	2.3%	5.1%
Fantasía	1.8%	3.5%	6.2%	1.6%	5.3%	6.2%	21.4%	1.4%	4.8%	4.1%	3.2%	7.6%	3.7%	2.5%	3.5%	5.3%	3.2%	6.0%	2.8%	5.8%
Filosofía	3.9%	0.7%	2.0%	11.1%	3.9%	2.6%	1.3%	44.4%	5.2%	2.0%	0.7%	2.0%	5.9%	3.3%	2.6%		2.0%	1.3%	0.7%	4.6%
Histórico	2.0%	8.8%	6.4%	3.2%	3.7%	5.1%	5.1%	1.9%	14.1%	4.4%	2.7%	6.1%	4.8%	3.1%	4.7%	5.1%	6.2%	4.6%	2.8%	5.4%
Humor	3.3%	3.3%	2.2%	2.2%	4.4%	4.9%	6.0%	1.6%	2.2%	35.0%	1.6%	5.5%	3.3%	0.5%	4.4%	4.4%	6.6%	3.8%	1.1%	3.8%
Intriga	1.9%	7.2%	5.1%	1.9%	5.4%	4.0%	5.6%	1.4%	2.7%	4.3%	17.8%	7.2%	2.5%	2.9%	4.3%	10.2%	3.2%	7.3%	2.2%	3.0%
Juvenil	2.6%	5.8%	4.2%	6.8%	4.2%	4.2%	1.6%	2.1%	2.6%	7.9%	4.2%	24.1%	5.2%	3.1%	4.2%	2.1%	3.7%	4.7%	1.6%	5.2%
Memorias	3.7%	3.7%	2.5%	6.8%	3.7%	9.9%	2.5%	6.2%	3.7%	4.3%	0.6%	7.5%	14.9%	1.2%	11.2%	2.5%	3.7%	3.7%	0.6%	6.2%
Novela	1.5%	5.4%	6.0%	1.2%	3.7%	5.8%	4.9%	0.8%	2.7%	3.5%	10.7%	7.5%	1.5%	11.4%	2.2%	8.7%	3.8%	11.6%	4.2%	2.8%
Otros	3.3%	6.2%	4.6%	6.5%	5.9%	3.3%	3.3%	4.6%	8.5%	4.2%	2.0%	2.9%	4.9%	2.0%	14.7%	3.9%	9.5%	1.3%	1.3%	7.2%
Policíaca	1.5%	5.6%	5.8%	2.3%	3.0%	3.8%	6.3%	0.8%	4.1%	5.6%	7.6%	4.3%	3.0%	2.5%	4.3%	22.5%	4.6%	3.5%	2.8%	6.1%
Realismo	3.6%	2.4%	1.2%	1.8%	8.4%	6.0%	4.2%	4.8%	4.2%	7.2%	1.2%	4.8%	6.6%	1.2%	7.8%	5.4%	21.1%	3.0%		4.8%
Romántico	1.4%	2.8%	2.3%	0.6%	3.8%	8.9%	5.6%	0.3%	1.9%	3.7%	3.4%	10.5%	1.8%	11.0%	1.8%	5.0%	3.2%	23.4%	4.2%	4.4%
Terror		9.5%	10.7%	0.2%	3.9%	1.7%	2.9%	1.5%	1.2%	2.2%	13.8%	3.2%	1.5%	0.5%	4.4%	9.5%	2.9%	3.2%	25.7%	1.7%
Variada	1.2%	5.1%	4.3%	3.1%	3.9%	6.6%	4.7%	3.1%	6.3%	5.5%	2.0%	4.7%	6.3%	2.7%	7.8%	9.0%	5.1%	5.1%	0.8%	12.9%

Figura 11: MS Experimento 2 Matriz de Confusión

Como es apreciable, este experimento, a pesar de obtener unos resultados de precisión numéricamente inferiores a los del primer experimento, es capaz de consistentemente clasificar correctamente la mayoría de los ejemplos de cada clase con la clase correspondiente, como es observable en la diagonal de verdaderos positivos.

A excepción de *Novela* (posiblemente la clase más general de todas) el resto tiene un porcentaje superior con verdaderos aciertos que ninguna otra clasificación errónea, demostrando que tener una base de datos balanceada afecta de manera muy notable el rendimiento de un clasificador. Se puede afirmar que aún con este avance el clasificador no es perfecto, pero sí que se observa que, a pesar de tener dificultades para discernir unas clases de otras, dado el hecho de que la función de pérdida se trata del error cuadrático medio (más apropiado para series numéricas), el porcentaje mayoritario de aciertos se concentra en los verdaderos positivos.

Experimento 3: 100 palabras comunes, datos balanceados, 20 clases (Cross Entropy)

Este experimento se realizó de manera adicional, buscando relacionar los resultados del Experimento 1 que utiliza la misma función de pérdida. Los datos son los mismos que los utilizados en el Experimento 2. El mejor valor también coincide con el seleccionado para el segundo experimento.

Overall accuracy	0.345221
Average accuracy	0.934522
Micro-averaged precision (F1)	0.345221
Macro-averaged precision	0.290225
Micro-averaged recall	0.345221
Macro-averaged recall	0.255349

Tabla 3: MS Experimento 3

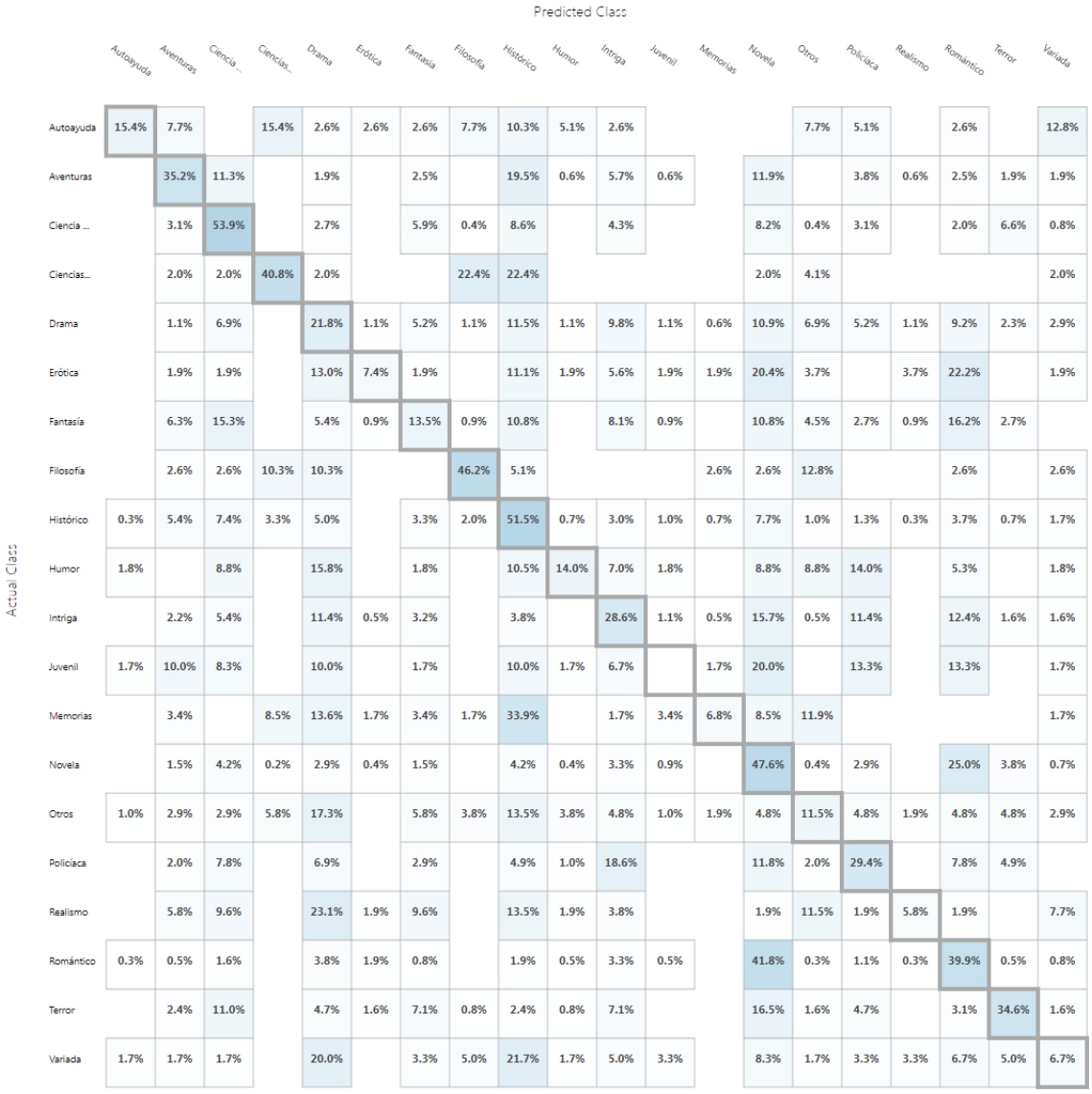


Figura 12: MS Experimento 3 Matriz de Confusión

Los resultados de este experimento muestran unos valores muy similares a los del primer experimento, si bien en cuanto a precisión numérica se han visto ligeramente reducidos. La

matriz de confusión es muy similar, teniendo conflictos en los mismos puntos en cuanto a falsos positivos para otras clases que tienen cierta relación.

Al no variar demasiado los resultados, se plantea que tal vez la presencia de representación equitativa en el set de datos no influya tanto a la hora de clasificar libros por su género, pero hay que destacar que este set de datos de entrenamiento era mucho menor que el utilizado en el primer experimento, y aún con un set superior de validación, el clasificador es capaz de aproximar los valores extremadamente bien, llegando a los niveles de precisión de otro clasificador con 3 veces su cantidad de ejemplos. Esto lleva a pensar que, si el set completo de datos estuviera más balanceado, tal vez se podrían obtener resultados más fiables para el clasificador, dada la capacidad de la red de extraer características de cada clase con un grado de confusión aceptable.

4.3.2 Resultados de Amazon Web Service

Amazon produce resultados en forma de fichero .csv, además de dar una pequeña vista previa en el propio servicio web. Estos ficheros .csv se pueden analizar para realizar una matriz de confusión similar a la producida por Microsoft, a fin de poder comparar ambos modelos de un modo más visual. La medida de precisión que proporciona Amazon utiliza el cómputo de F1, una función que estima la precisión con un valor entre 0 y 1 según los falsos positivos.

Experimento 1: 100 palabras comunes, datos sin balancear, 20 clases, parámetros por defecto

El primer experimento realizado emplea el clasificador por defecto de Amazon, con razones de aprendizaje en el rango 0.01 – 100, número de iteraciones máximo de 10, regularización L2 de tipo Suave (1e-6) y sin aleatorizar el orden de los datos. El modelo resultante es un modelo con razón de aprendizaje 1, 10 iteraciones de aprendizaje y rendimiento en F1 como se indica en la tabla:

Evaluación	F1
Media	0.197
Base	0.015

Tabla 4: AWS Experimento 1

	Novela	omántic	ncia ficc	istóric	Terror	Intriga	ventura	Drama	Fantasía	Otros	Policíaca	Juvenil	Humor	Variada	cias soc	ealism	utoayuc	Erótica	emoria	ilosofía
Novela	32.5%	19.3%	17.1%	4.1%	0.8%	11.6%	3.0%	2.4%	1.2%	0.2%	7.5%	0.0%	0.2%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Romántico	43.8%	41.7%	1.2%	1.5%	0.6%	2.7%	0.3%	3.0%	1.8%	0.0%	2.7%	0.6%	0.0%	0.0%	0.0%	0.0%	0.0%	0.3%	0.0%	0.0%
Ciencia ficción	13.1%	2.6%	44.1%	7.0%	1.0%	11.5%	2.6%	2.6%	4.8%	0.6%	9.3%	0.0%	0.0%	1.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Histórico	9.1%	4.7%	12.8%	50.0%	0.7%	1.7%	6.1%	4.1%	1.7%	1.0%	2.7%	0.3%	0.0%	1.0%	2.7%	0.3%	0.0%	0.3%	0.3%	0.3%
Terror	23.2%	3.9%	13.3%	3.3%	1.1%	32.6%	1.7%	2.8%	2.8%	0.0%	14.9%	0.0%	0.0%	0.6%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Intriga	18.2%	10.9%	9.1%	6.7%	0.6%	26.7%	0.6%	6.7%	2.4%	0.6%	15.8%	0.0%	0.0%	0.6%	0.0%	0.6%	0.0%	0.0%	0.6%	0.0%
Aventuras	18.6%	4.5%	25.0%	18.6%	0.0%	4.5%	8.3%	5.1%	4.5%	0.6%	8.3%	0.0%	0.6%	0.6%	0.0%	0.6%	0.0%	0.0%	0.0%	0.0%
Drama	12.3%	10.9%	2.9%	15.9%	1.4%	8.7%	2.2%	24.6%	5.8%	2.9%	8.7%	0.0%	0.0%	2.2%	0.0%	0.0%	0.0%	0.7%	0.7%	0.0%
Fantasía	18.3%	13.3%	10.8%	16.7%	1.7%	8.3%	1.7%	4.2%	14.2%	0.8%	6.7%	0.0%	0.0%	0.8%	0.0%	0.8%	0.0%	0.8%	0.0%	0.8%
Otros	7.0%	4.0%	14.0%	13.0%	2.0%	8.0%	3.0%	20.0%	4.0%	10.0%	3.0%	0.0%	0.0%	1.0%	2.0%	3.0%	0.0%	0.0%	1.0%	5.0%
Policíaca	17.2%	10.3%	11.5%	3.4%	1.1%	24.1%	1.1%	3.4%	3.4%	0.0%	23.0%	0.0%	0.0%	1.1%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Juvenil	17.5%	8.8%	22.5%	12.5%	1.3%	8.8%	10.0%	5.0%	3.8%	2.5%	6.3%	0.0%	1.3%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Humor	21.3%	11.5%	8.2%	11.5%	0.0%	11.5%	0.0%	16.4%	3.3%	0.0%	6.6%	0.0%	4.9%	1.6%	0.0%	0.0%	0.0%	1.6%	1.6%	0.0%
Variada	16.9%	11.9%	3.4%	16.9%	0.0%	8.5%	0.0%	23.7%	0.0%	1.7%	6.8%	0.0%	0.0%	1.7%	1.7%	0.0%	0.0%	0.0%	3.4%	3.4%
Ciencias sociales	3.6%	1.8%	3.6%	26.8%	0.0%	0.0%	0.0%	3.6%	0.0%	10.7%	0.0%	0.0%	0.0%	0.0%	32.1%	0.0%	0.0%	0.0%	3.6%	14.3%
Realismo	8.1%	5.4%	2.7%	16.2%	0.0%	10.8%	0.0%	40.5%	8.1%	2.7%	2.7%	0.0%	0.0%	2.7%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Autoayuda	5.9%	8.8%	2.9%	11.8%	2.9%	8.8%	5.9%	8.8%	2.9%	0.0%	5.9%	0.0%	5.9%	2.9%	8.8%	0.0%	2.9%	2.9%	0.0%	11.8%
Erótica	39.4%	24.2%	6.1%	3.0%	0.0%	3.0%	3.0%	6.1%	3.0%	0.0%	6.1%	0.0%	0.0%	3.0%	0.0%	0.0%	0.0%	3.0%	0.0%	0.0%
Memorias	6.3%	3.1%	3.1%	43.8%	0.0%	6.3%	3.1%	12.5%	0.0%	0.0%	3.1%	0.0%	0.0%	3.1%	0.0%	0.0%	0.0%	0.0%	12.5%	3.1%
Filosofía	4.0%	8.0%	4.0%	8.0%	0.0%	4.0%	0.0%	16.0%	0.0%	8.0%	0.0%	0.0%	0.0%	0.0%	8.0%	0.0%	0.0%	0.0%	0.0%	40.0%

Figura 13: AWS Experimento 1 Matriz de Confusión

De acuerdo con la página de resultados del experimento en, la puntuación de este modelo supera a la puntuación base, pero realmente al inspeccionar los datos vemos que la precisión como tal deja mucho que desear en este clasificador.

Aun así, las clases que tienen el mayor índice de falsos positivos confunden clases relacionadas con la clase real. La organización de las clases pasa de ser por orden alfabético como lo era en los experimentos de Microsoft y pasa a ser por orden de aparición, y resulta interesante observar como la mayoría de las predicciones, tanto erróneas como acertadas, caen dentro de la esquina inferior izquierda de la tabla, bajo la diagonal de verdaderos positivos. Si bien las confusiones que se dan coinciden con aquellas de los experimentos anteriores, es extraño ver que la organización por orden de aparición es tan acertada en cuanto a la confusión con clases similares, distribuyendo las clasificaciones de una manera tan visual.

Experimento 2: 100 palabras comunes, datos sin balancear, 20 clases, parámetros propios

En el segundo experimento se probó a modificar los parámetros por defecto que trae el clasificador, para observar cómo afectan estos a la clasificación de los datos. Las razones de aprendizaje coinciden con las del Experimento 1, pero el máximo número de iteraciones aumenta a 20, el algoritmo de regularización sigue siendo L2, pero con valor Medio (1e-4). El modelo generado tiene una razón de aprendizaje de 10 y 20 iteraciones. El rendimiento en F1 medio de este clasificador es de:

Evaluación	F1
Media	0.233
Base	0.014

Tabla 5: AWS Experimento 2

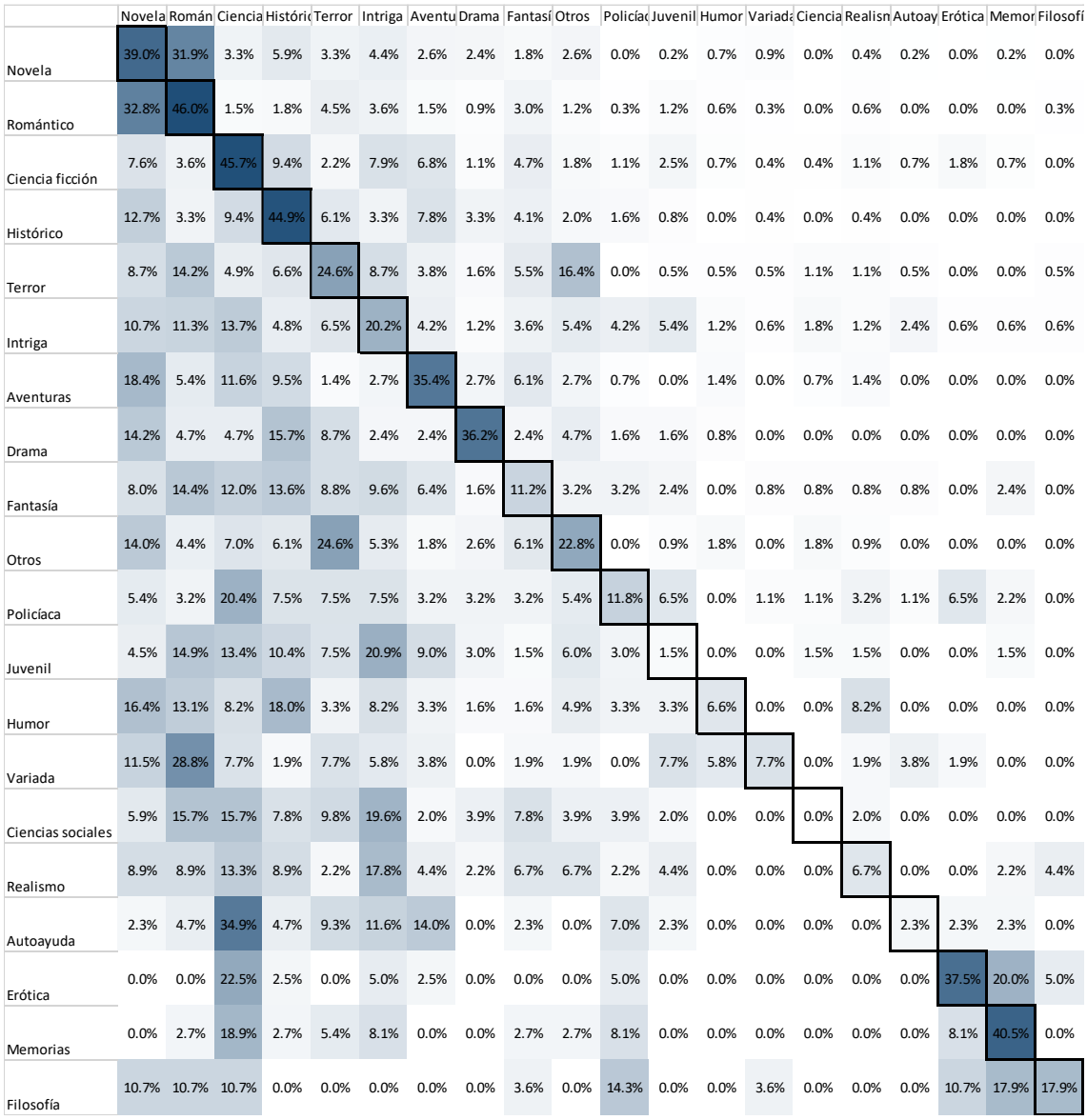


Figura 14: AWS Experimento 2 Matriz de Confusión

En este experimento se ve una notable mejora respecto al primero, tanto en el valor de F1 como en los resultados de la matriz de confusión. Las clases más similares siguen presentando

confusión, como el caso de *Novela* y *Romántico*, que componen más del 70% de las clasificaciones de ambas clases, como falsos y verdaderos positivos, pero a pesar de ello se nota un incremento en la precisión de manera general, no se elimina la confusión, pero sí se disminuye su efecto. Se estima que este cambio tan notable se debe al aumento de la regularización de L2, creando una matriz más estable con una razón de aprendizaje mucho mayor (10).

Experimento 3: 100 palabras comunes, datos balanceados, 20 clases, parámetros por defecto

El tercer experimento cambia de set de datos para utilizar los datos balanceados con 2000 instancias y ejemplos con la misma representación en todo el set. Los parámetros son los que ofrece Amazon por defecto, por lo que equivalen a los del Experimento 1. El modelo generado tiene razón de aprendizaje 1 y 10 iteraciones. El rendimiento en F1 es de:

Evaluación	F1
Media	0.379614
Base	0.002

Tabla 6: AWS Experimento 3

	Novela	Román	Histórico	Ciencia	Intriga	Drama	Aventu	Fantasí	Terror	Policíac	Otros	Variada	Juvenil	Humor	Erótica	Realism	Ciencia	Memor	Filosofí	Autoayu
Novela	16.0%	20.1%	1.4%	8.5%	5.2%	5.2%	5.2%	2.9%	3.1%	7.6%	1.3%	2.4%	5.8%	3.3%	6.8%	2.5%	0.2%	0.8%	0.3%	1.2%
Romántico	13.0%	37.4%	0.4%	1.8%	2.6%	4.2%	1.3%	3.8%	2.4%	5.7%	0.7%	2.6%	9.0%	2.1%	9.5%	2.0%	0.0%	0.4%	0.1%	1.0%
Histórico	0.9%	1.5%	27.5%	8.6%	0.6%	7.8%	7.1%	2.7%	0.8%	3.0%	2.7%	4.1%	3.6%	3.8%	3.6%	6.3%	6.0%	6.5%	1.9%	0.9%
Ciencia ficción	1.2%	2.2%	5.0%	30.8%	5.5%	6.8%	10.0%	2.9%	1.1%	6.7%	3.4%	4.6%	5.4%	3.2%	2.3%	3.6%	0.5%	2.2%	1.4%	1.3%
Intriga	2.4%	5.9%	1.6%	8.4%	19.7%	7.3%	6.5%	4.0%	1.4%	17.3%	1.1%	5.2%	6.2%	2.5%	3.7%	4.1%	0.0%	1.6%	0.2%	0.8%
Drama	1.9%	5.7%	2.8%	3.6%	2.4%	26.1%	3.1%	2.3%	1.3%	5.5%	6.3%	7.6%	5.0%	5.0%	4.9%	9.2%	0.2%	4.4%	1.0%	1.8%
Aventuras	1.1%	1.9%	7.4%	11.7%	1.7%	7.4%	29.9%	2.8%	1.3%	5.5%	3.6%	2.6%	4.3%	3.8%	2.6%	6.8%	0.6%	2.8%	0.6%	1.5%
Fantasía	1.4%	4.6%	6.5%	6.2%	2.1%	6.9%	4.6%	27.6%	0.5%	7.6%	2.8%	4.1%	7.1%	3.5%	5.8%	4.4%	0.2%	2.5%	0.7%	0.9%
Terror	1.5%	1.0%	1.2%	10.9%	20.1%	6.3%	2.9%	1.9%	26.2%	9.7%	2.2%	3.2%	2.7%	1.5%	1.2%	4.6%	0.5%	1.0%	1.0%	0.5%
Policíaca	1.5%	4.1%	0.3%	7.8%	7.1%	6.6%	2.3%	2.0%	1.3%	42.8%	2.0%	3.5%	7.3%	2.8%	3.0%	4.3%	0.0%	0.8%	0.3%	0.3%
Otros	0.3%	1.6%	2.0%	6.9%	0.3%	9.2%	2.3%	1.0%	0.0%	1.3%	41.8%	3.9%	3.9%	3.9%	2.9%	2.6%	4.6%	3.3%	5.2%	2.9%
Variada	0.4%	5.9%	4.7%	3.5%	1.2%	6.6%	0.8%	2.0%	0.0%	3.9%	2.7%	43.0%	3.5%	2.7%	7.8%	4.3%	2.0%	2.7%	0.8%	1.6%
Juvenil	3.1%	3.7%	2.1%	2.1%	1.0%	3.7%	6.8%	3.1%	0.0%	3.1%	4.2%	1.6%	58.1%	4.2%	1.6%	0.0%	0.0%	1.0%	0.0%	0.5%
Humor	0.5%	1.1%	2.7%	4.4%	1.6%	3.3%	2.7%	1.1%	0.0%	6.0%	1.6%	1.6%	3.8%	60.1%	4.4%	1.1%	0.0%	2.2%	0.0%	1.6%
Erótica	4.0%	9.6%	1.1%	0.6%	0.6%	4.0%	0.0%	0.6%	0.6%	1.7%	0.6%	1.1%	3.4%	1.7%	66.1%	2.3%	0.0%	1.1%	0.0%	1.1%
Realismo	1.2%	1.2%	3.6%	1.8%	0.0%	5.4%	1.2%	1.2%	0.0%	3.0%	2.4%	1.2%	3.0%	2.4%	2.4%	66.3%	0.6%	2.4%	0.0%	0.6%
Ciencias sociales	0.6%	0.6%	1.8%	0.0%	0.0%	2.5%	0.0%	0.0%	0.0%	0.0%	1.2%	0.0%	0.0%	0.0%	0.0%	0.6%	80.4%	2.5%	7.4%	2.5%
Memorias	0.6%	1.9%	3.7%	1.2%	0.0%	3.1%	0.6%	0.0%	0.0%	1.2%	2.5%	1.2%	0.0%	1.2%	2.5%	2.5%	1.9%	69.6%	2.5%	3.7%
Filosofía	0.0%	0.0%	0.0%	0.7%	1.3%	2.0%	0.0%	0.0%	0.0%	0.0%	1.3%	1.3%	0.7%	0.0%	0.7%	0.0%	7.8%	0.7%	83.0%	0.7%
Autoayuda	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	100.0%

Figura 15: AWS Experimento 3 Matriz de Confusión

Hasta el momento, este experimento es el que ha rendido los mejores resultados en F1, y en la matriz de confusión podemos observar el porqué: los datos "convergen" según menos ejemplos tenemos de cada clase en el set de validación, teniendo una de precisión elevada para aquellos ejemplos escasos. Este resultado se corresponde con un experimento que ha utilizado una serie

de ejemplos repetidos para validación. Lo que hace sospechar es la existencia de ejemplos de clasificación en clases como *Autoayuda*, que cuenta con 100 ejemplos en total, todos presentes en el set de entrenamiento, y que figuran en la matriz de confusión, cuando el valor de toda la fila debería ser de 0, al no existir ejemplos a clasificar de esa clase. Dado que el fichero de validación no contiene ningún ejemplo repetido (se generó utilizando un script de eliminación de ejemplos duplicados), cabe pensar que el clasificador está tomando los aciertos en el entrenamiento en cuenta para valorar la precisión del clasificador.

Si esa hipótesis fuese cierta, este experimento no tendría validez más que para las clases con muchas más de 100 instancias, ya que para aquellas con justo el número base la precisión se dispara.

Experimento 4: palabras comunes, datos balanceados, 20 clases, parámetros propios

Al igual que el tercer experimento, este utiliza el set de datos balanceados para entrenar y el completo sin las instancias de entrenamiento como validación. El número de iteraciones es 20, la función de normalización es L2 del orden Agresivo (1e-2). La red generada tiene razón de aprendizaje 1 y 20 iteraciones realizadas. El valor de F1 es:

Evaluación	F1
Media	0.395382
Base	0.002

Tabla 7: AWS Experimento 4

	Novela	Román	Histórico	Ciencia	Intriga	Drama	Aventu	Fantasí	Terror	Policíac	Otros	Variada	Juvenil	Humor	Erótica	Realism	Ciencia	Memor	Filosofí	Autoayu
Novela	17.8%	21.0%	1.8%	8.5%	6.1%	2.2%	6.9%	1.8%	2.9%	9.8%	1.4%	2.0%	6.0%	1.8%	5.8%	2.4%	0.3%	0.7%	0.5%	0.4%
Romántico	15.5%	40.8%	0.4%	0.6%	2.9%	1.2%	2.0%	3.1%	3.0%	6.8%	0.7%	1.9%	7.7%	2.2%	8.6%	1.9%	0.1%	0.2%	0.1%	0.4%
Histórico	1.0%	1.5%	33.1%	8.1%	0.5%	2.3%	9.2%	1.9%	0.9%	4.0%	3.1%	4.2%	2.7%	1.8%	3.8%	5.6%	7.2%	7.1%	1.8%	0.2%
Ciencia ficción	0.7%	1.6%	6.4%	32.4%	4.9%	4.6%	11.1%	2.1%	0.9%	10.7%	3.7%	3.4%	5.0%	2.2%	1.8%	3.3%	0.8%	1.8%	1.6%	0.8%
Intriga	2.7%	5.9%	2.7%	10.3%	19.6%	3.8%	7.2%	2.4%	1.7%	20.8%	2.1%	3.5%	5.7%	2.4%	2.1%	4.8%	0.0%	1.4%	0.3%	0.6%
Drama	1.6%	6.2%	3.6%	2.9%	2.6%	22.9%	4.4%	1.6%	1.1%	7.6%	7.5%	6.5%	5.8%	4.1%	5.2%	8.3%	1.1%	4.5%	1.5%	1.1%
Aventuras	1.1%	1.1%	9.8%	13.6%	2.1%	1.7%	31.9%	3.2%	1.5%	7.4%	4.3%	2.6%	4.2%	3.6%	1.7%	5.1%	0.8%	3.2%	0.6%	0.4%
Fantasía	1.4%	3.5%	6.5%	8.8%	1.4%	2.3%	6.9%	26.3%	0.9%	10.8%	4.8%	2.8%	7.6%	2.5%	4.6%	4.8%	0.5%	2.3%	0.9%	0.5%
Terror	1.0%	0.2%	1.7%	15.3%	17.7%	3.4%	3.6%	1.2%	25.0%	9.2%	3.4%	2.2%	2.9%	2.2%	1.7%	6.1%	0.7%	1.2%	1.0%	0.2%
Policíaca	0.8%	4.1%	1.0%	9.1%	7.1%	1.8%	2.3%	2.3%	1.8%	51.4%	1.8%	2.0%	5.3%	2.3%	1.8%	4.1%	0.0%	0.8%	0.3%	0.3%
Otros	0.3%	0.3%	3.6%	3.6%	1.0%	4.9%	3.9%	0.7%	0.3%	4.6%	43.1%	2.6%	2.3%	3.9%	1.6%	3.6%	6.2%	4.2%	6.9%	2.3%
Variada	0.4%	7.0%	3.5%	2.0%	0.8%	3.5%	1.2%	2.0%	0.0%	3.5%	4.3%	43.0%	3.5%	2.3%	7.4%	6.3%	3.1%	3.1%	2.0%	1.2%
Juvenil	4.2%	3.1%	3.1%	3.1%	0.5%	0.5%	7.9%	3.1%	1.0%	5.2%	3.7%	1.0%	57.6%	3.1%	1.6%	1.0%	0.0%	0.0%	0.0%	0.0%
Humor	1.1%	2.2%	2.2%	3.8%	2.2%	1.6%	4.4%	0.0%	0.0%	6.6%	1.6%	1.6%	3.3%	61.2%	2.2%	2.7%	0.0%	2.2%	0.0%	1.1%
Erótica	4.5%	9.6%	1.1%	2.3%	0.0%	1.7%	1.7%	0.0%	0.6%	2.3%	1.7%	1.1%	2.8%	2.3%	65.5%	1.7%	0.6%	0.0%	0.6%	0.0%
Realismo	1.2%	1.2%	3.6%	1.2%	0.6%	3.0%	4.8%	1.8%	0.0%	2.4%	2.4%	3.6%	2.4%	2.4%	1.8%	65.7%	0.6%	0.6%	0.0%	0.6%
Ciencias sociales	0.0%	0.6%	1.2%	0.0%	0.0%	1.2%	0.6%	0.0%	0.0%	0.0%	1.2%	0.0%	0.0%	0.6%	0.0%	0.0%	83.4%	1.2%	8.0%	1.8%
Memorias	0.6%	1.2%	3.1%	1.9%	0.0%	0.6%	1.2%	0.0%	0.0%	0.6%	3.7%	1.2%	0.0%	1.2%	2.5%	2.5%	2.5%	71.4%	3.1%	2.5%
Filosofía	0.0%	0.7%	0.0%	0.7%	1.3%	0.7%	0.0%	0.0%	0.0%	0.0%	1.3%	0.7%	0.7%	0.0%	0.7%	0.0%	8.5%	0.7%	83.7%	0.7%
Autoayuda	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	100.0%

Figura 16: AWS Experimento 4 Matriz de Confusión

Al igual que el experimento anterior, este proporciona unos resultados que tienen un aspecto poco creíble, debido a las clases con menos ejemplos. Por desgracia, poca información es extraíble de estos experimentos que, pese a tener una buena base y haber funcionado en el sistema de Microsoft, fallan al ejecutarse con Amazon.

4.3.3 Resultados del Perceptrón Multicapa

La implementación local de la red neuronal obtuvo resultados menos visuales que las provistas por los servicios en red dado que la salida del programa de entrenamiento y validación no generaba más datos que la precisión del modelo. Existe una forma de extraer datos del modelo entrenado, pero por cuestiones de tiempo fue imposible realizar matrices de confusión o evaluaciones con F1 para ser capaces de comparar directamente con los otros modelos.

La estructura de la red es invariable, con 2 capas ocultas y 100 nodos por capa oculta, utilizando de función de pérdida la función SoftMax y de función de optimización el algoritmo Adam. Las iteraciones cubren el rango {10,25,50,75,100} y las razones de aprendizaje (r.A.) {0.001,0.01,0.1,1}.

Debido a estos contratiempos, se presentan las medias recabadas para cada experimento en la siguiente tabla:

Set de datos	Iteraciones,r.A.	Precisión
Texto completo (tratado)	10,0.01	0.259
100 palabras comunes	25,0.01	0.347
100 palabras comunes balanceadas	10,0.1	0.316

Tabla 8: Perceptrón Experimentos

Dada la simplicidad del modelo, no es de extrañar que la precisión del mismo no supere el 40% en el mejor de los casos. Al utilizar una medida tan sencilla como la frecuencia de aparición de términos en el texto, la capacidad de distinción de clases no dista mucho de la que podría conseguir un clasificador como un árbol de decisión. Sin embargo, tras la realización de los experimentos en los servicios web, es interesante ver como la utilización de las 100 palabras más comunes para la clasificación supera con creces el único modelo entrenado utilizando la totalidad del texto (previo tratamiento). Esto demuestra una vez más que la calidad de los datos puede determinar la calidad de los resultados obtenidos, independientemente del modelo empleado para ello.

Al utilizar el Perceptrón como medida mínima para comparar los otros dos sistemas de clasificación, resulta menos dañino para el cumplimiento de los objetivos no poder analizar los datos correspondientes a este modelo en contraposición con los de los servicios web.

4.4 Análisis comparativo de los resultados

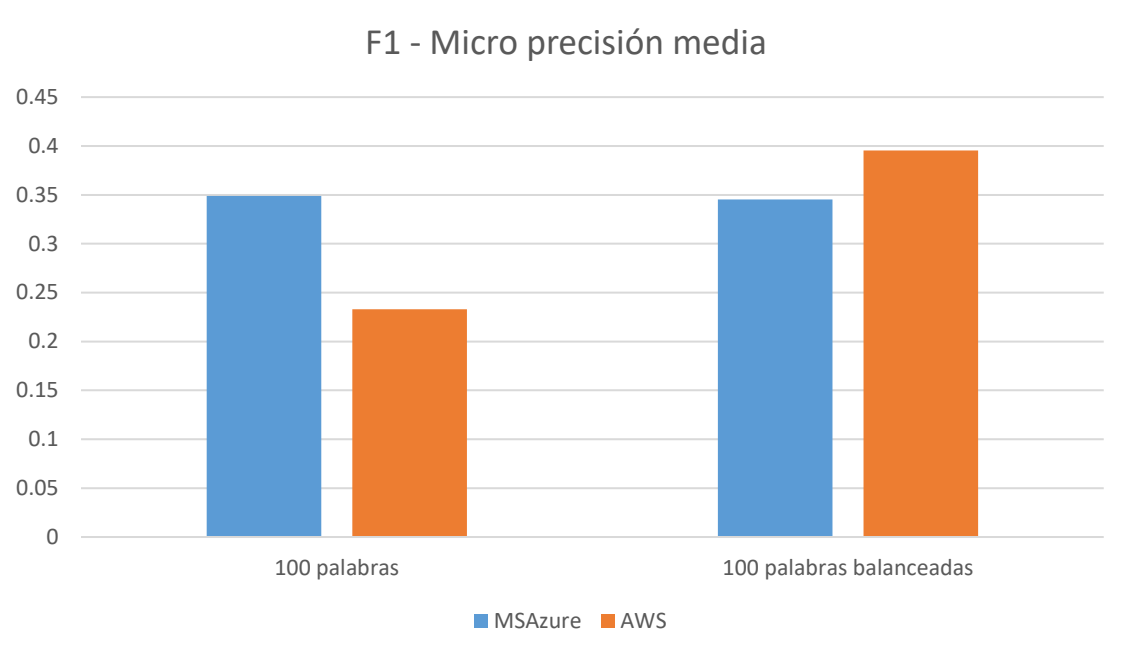


Gráfico 2: F1 - Micro precisión media

A pesar de la representación del gráfico, los resultados de los experimentos nos dan a entender que el servicio de Amazon utilizó datos repetidos para clasificar sus instancias menos pobladas, por lo que la categoría de datos balanceados queda como intratable a la hora de sacar conclusiones de los resultados.

Por otra parte, en los resultados obtenidos de la clasificación con las 100 palabras más comunes, utilizando el 70% aleatorio de los datos de entrada como entrenamiento y el 30% restante como validación, se observa claramente que el modelo de Microsoft tiene ventaja en cuanto a precisión de clasificación. Aun así, ambos son realmente buenos a la hora de establecer relaciones entre clases similares, como se puede observar en la tabla:

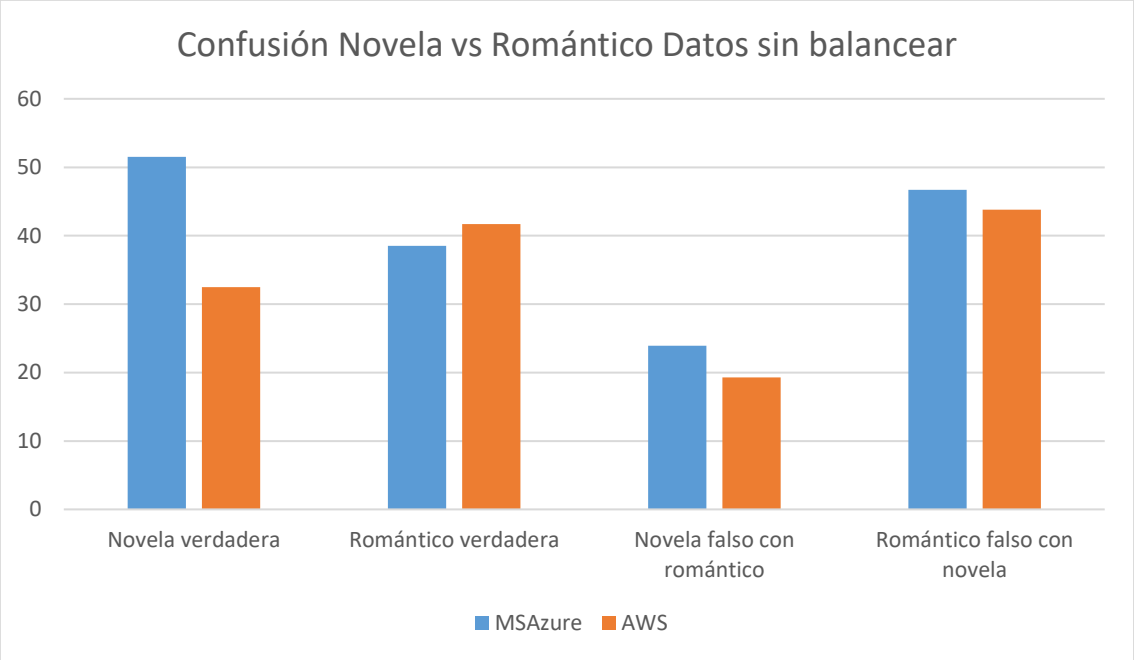


Gráfico 3: Confusión Novela vs Romántico Datos sin balancear

Analizando las matrices de confusión de cualquiera de los dos clasificadores para datos sin balancear existen diversos ejemplos de clases que tienen una distribución de aciertos y falsos positivos similar. Es innegable que existe una relación de parecido entre la clase denominada *Novela* y la clase *Romántico*, dado que una clase tan general como *Novela* tiene altas probabilidades de contener elementos que sean de algún modo parte de una obra romántica, y del mismo modo una obra de estilo *Romántico* por norma general se clasifica también como *Novela*, cerrando el ciclo.

Este tipo de confusiones se da por la similitud de los géneros literarios entre sí, no existe un verdadero umbral que los distinga al 100%. Por ello sería digno de consideración la inclusión de un campo denominado “segunda clase” o “género secundario”, para poder clasificar con un segundo puesto para la clase más similar a la clasificada, potencialmente reduciendo los falsos positivos de manera drástica.

Capítulo 5: Impacto social y económico

Este capítulo analiza la repercusión de los resultados obtenidos tras este estudio, y las posibles aplicaciones de los mismos en el mundo real. Además, se establece el coste aproximado del desarrollo de este proyecto y se estima el coste del desarrollo de un sistema completo utilizando el sistema de clasificación diseñado.

5.1 Repercusión del proyecto

Dados los resultados obtenidos tras las pruebas cabe destacar que la repercusión del desarrollo de este estudio no será notable en el mundo real, al menos si no se realiza un informe que comunique a las compañías responsables de los sistemas empleados de los inconvenientes y posibles deficiencias de sus modelos en red. Por lo tanto, se considera que el impacto en cuanto a mejora del rendimiento de los sistemas de clasificación en la nube de las compañías estudiadas será nulo.

Sin embargo, en cuanto a clasificación literaria se refiere, este estudio puede arrojar luz sobre las similitudes y puntos de confusión entre los distintos géneros literarios considerados, dando pie a continuar una línea de investigación para la mejora de la clasificación de libros, al menos en la lengua castellana.

5.1.1 Posibles aplicaciones

Tanto con el uso del clasificador implementado en TensorFlow como los servicios en red empleados se podría realizar un clasificador de libros en castellano en un tiempo relativamente reducido, que puede servir para resolver tareas de clasificación como las del caso supuesto en la motivación de este estudio, un sistema clasificador para recomendación de concursos literarios a los autores de las obras clasificadas. La calidad del clasificador no está en condiciones de prestar un servicio completo a los usuarios, pero con la inclusión de un sistema corrector ortográfico, la extracción de palabras ajenas al vocabulario para su inclusión en un sistema de *stemming* especializado y algún método de reducción del texto a un tamaño más tratable, el clasificador resultante podría ofrecer resultados como los obtenidos en estudios similares.

Además de un clasificador para un sistema de recomendación, este tipo de clasificadores puede ser útil en la inclusión de nuevos títulos en librerías y bibliotecas, incluso puede servir de sistema de apoyo en empresas de publicación de libros, realizando la clasificación de los libros a publicar a modo de verificador del estilo empleado. Incluso se podría utilizar este sistema clasificador para validar textos generados por máquinas inteligentes que emulen la escritura humana, como el bot de Deep Learning que genera textos de Shakespeare (Karpathy, 2015).

5.2 Recursos empleados

Este apartado analizará los recursos empleados a lo largo del estudio con el fin de generar un presupuesto para la realización del mismo. Se incluirá en la recopilación de costes todo aquello que genere un gasto monetario, asociado a su coste estimado.

5.2.1 Fases del proyecto

Para mejor desglosar los recursos empleados se dividirá el proyecto en las distintas fases que lo componen, indicando el papel desempeñado durante el tiempo de duración de cada fase, para ajustar el coste de personal a cada tarea realizada, teniendo en cuenta para el cómputo el nivel de formación necesario para llevar a cabo cada tarea.

Fases	Duración de la fase															
	MARZO				ABRIL				MAYO				JUNIO			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
F1 – Propuesta de trabajo																
F2 – Planificación y organización																
F3 – Análisis																
F4 – Diseño																
F5 – Implementación																
F6 – Pruebas y evaluación																
F7 – Presentación del proyecto																

Tabla 9: Fases del Proyecto

5.2.2 Desglose del presupuesto

En función de las distintas fases del proyecto, el coste del desempeño de cada fase es variante en cuanto a valor monetario. Para calcular un valor aproximado se han utilizado los datos actuales del salario medio de un profesional cualificado para llevar a cabo cada tarea.

Fases	Jefe de Proyecto	Analista	Diseñador	Desarrollador	Tester
F1	2 semanas				
F2	1 semana				
F3		4 semanas			
F4			3 semanas		
F5				3 semanas	
F6					4 semanas
F7	2 semanas				
Total	5 semanas	4 semanas	3 semanas	3 semanas	4 semanas

Tabla 10: Tiempo de trabajo de cada función

Tomando como una semana laborable trabajando en el proyecto a media jornada, 4 horas diarias rinde un coste de personal de:

Fases	Jefe de Proyecto	Analista	Diseñador	Desarrollador	Tester
F1 (€)	1,480				
F2 (€)	725				
F3 (€)		2,240			
F4 (€)			1,680		
F5 (€)				1,500	
F6 (€)					1,600
F7 (€)	1,480				
Total (€)	**3,685**	**2,240**	**1,680**	**1,500**	**1,600**

Tabla 11: Tabla de costes de personal por tiempo trabajado

El uso de equipos para el desarrollo del proyecto, el software utilizado y sus licencias, así como los costes impuestos por las empresas para el uso de sus servicios online rinden un coste de:

Descripción	Unidades	Coste / unidad (€)
Ordenador portátil *New Razer Blade Stealth*	1	1,700
Ubuntu 17.04	1	0

Python 2.7	1	0
Librería TensorFlow	1	0
Microsoft Office 365 Personal	1	6.99 / mes
Licencia estándar Microsoft Azure Machine Learning	1	0
Licencia estándar Amazon Web Services	1	0

Tabla 12: Costes de equipo y licencias

Los gastos fungibles junto con los gastos mensuales se detallan en la siguiente tabla:

Concepto	Coste / mes (€)
Material de Oficina	10
Luz	56.3
Agua	30
Conexión a Internet	70
Transporte	20
Total	**186.3**

Tabla 13: Costes adicionales

5.4 Resumen del presupuesto

Realizando un cómputo total del presupuesto desglosado en el apartado anterior, estimamos el coste total del desarrollo del proyecto de investigación en:

Concepto	Coste (€)
Personal	10,705
Material	1,727.96
Otros	745.2
Total	**13,178.16**

Tabla 14: Resumen del presupuesto

Capítulo 6: Conclusiones y líneas futuras

El último capítulo abarca las conclusiones obtenidas a través de la realización del estudio con respecto de los objetivos principales marcados al comienzo del mismo, especula sobre posibles líneas de trabajo futuras que permitan que los hallazgos de este proyecto continúen aportando beneficio, relata los problemas encontrados en la búsqueda de soluciones y la implementación de las mismas y finaliza con las conclusiones personales sobre el proyecto y el campo de estudio en sí.

6.1 Conclusiones respecto a los objetivos

Como ha quedado patente a lo largo del estudio, los sistemas de clasificación en la nube proporcionados por Microsoft y Amazon son perfectamente capaces de tratar datos, generar modelos de clasificación y utilizar los datos tratados para entrenar y validar los modelos generados. No por ello automáticamente pasan a ser modelos válidos para llevar a un entorno de producción, primero ha de comprobarse que superan un mínimo de precisión para poder confiar en sus predicciones.

El Perceptrón Multicapa implementado presenta numerosos problemas en cuanto a diseño y adaptabilidad a este problema, si bien es considerado un aproximador universal, esta red no está pensada para realizar la tarea de clasificación de textos con la mayor de las precisiones. Es por ello que queda constancia del bajo rendimiento de esta red en este entorno particular. Por tanto, se considera que el objetivo de implementar un Perceptrón Multicapa como medida base para la comparativa con las redes de servicios web no queda cumplido en su totalidad, ya que, aunque implementado, no cumple su función comparativa.

Por otra parte, los sistemas web estudiados proveen resultados prometedores en el campo de la clasificación literaria, si bien no proporcionan resultados perfectos ni mucho menos, pero consiguen relacionar los libros con clases correctas o cercanamente relacionadas con la clase verdadera. El empleo de una segunda clase podría mejorar las ocurrencias de falsos positivos, pero un sistema de clasificación que elija dos clases en función de probabilidad quedaba fuera del alcance de este proyecto. Por lo tanto, se considera cumplido el objetivo de discernir si los servicios web son útiles para clasificación de textos, y como respuesta se da una afirmación, con la condición de que el sistema que reciba la clasificación acepte una segunda clase, una vez modelada la solución apropiada.

6.2 Líneas de trabajo futuras

Como se ha descrito en las conclusiones, la ampliación de los modelos desarrollados para acomodar una segunda clase a la clasificación de los libros sería el primer paso para afianzar la calidad de la clasificación respecto de los géneros más similares entre sí. Pasado este punto, la clasificación de textos no sólo implementa los recursos detallados en este estudio, sino que añade más dimensiones al estudio de los textos.

Técnicas como el "embedding" son populares a la hora de extraer la temática o la intención de un texto, y pueden arrojar luz al sistema desarrollado en este proyecto. La utilización de esta técnica implicaría el "posicionamiento" de cada palabra en un grafo que permite visualizar la distancia matemática de una palabra con el resto de las figurantes en el vocabulario, significando que una palabra posee una tendencia mayor o menor a aparecer en conjunto con N otras, siendo N el tamaño de la ventana de observación para cada palabra y sus adyacentes.

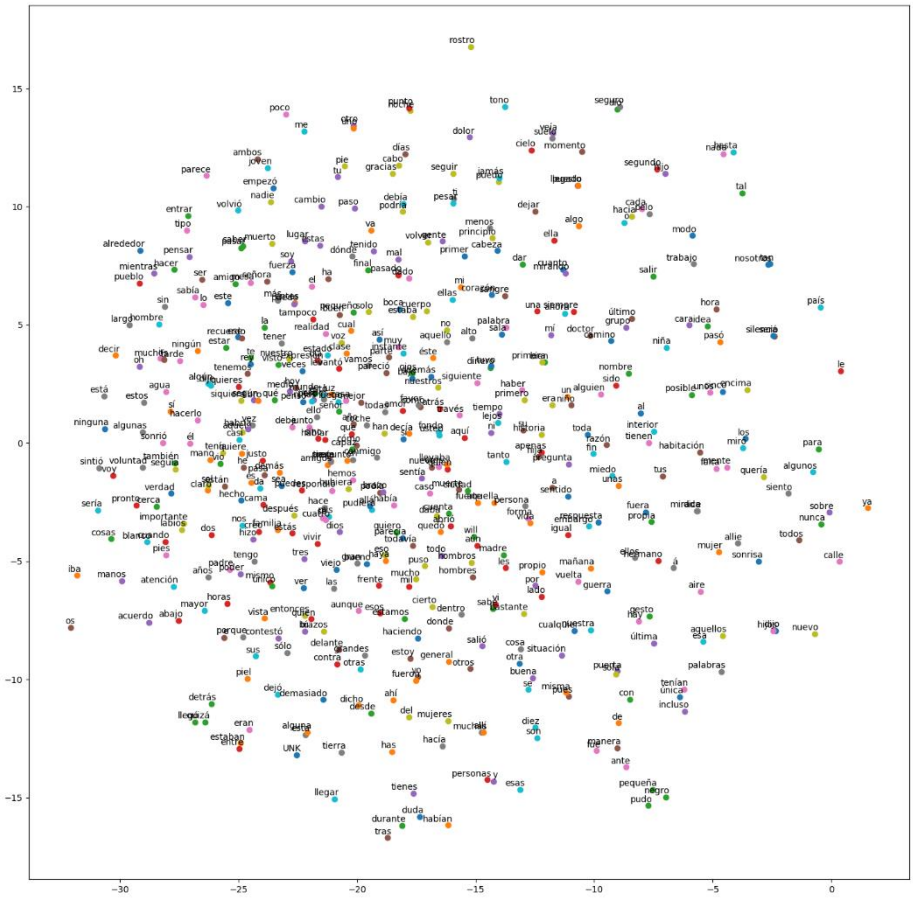


Figura 17: Gráfico de embeddings

Como se ha mencionado en varias ocasiones en el documento, la motivación de este estudio reside en la necesidad de un sistema de clasificación de libros para el apoyo en la clasificación

de libros escritos por potenciales participantes en concursos literarios a través de una plataforma online. La implementación de un sistema de clasificación en un entorno controlado puede ampliar la base de datos de entrenamiento del modelo, mejorando con cada ejemplo nuevo su modelo aprendido.

6.3 Problemas encontrados

El proyecto no ha estado libre de problemas y contratiempos, comenzando por el enfoque del proyecto en sí. Lo que comenzó siendo una tarea de diseño e implementación de un sistema de clasificación mediante redes de neuronas artificiales se convirtió en un proyecto de estudio comparativo de las distintas tecnologías existentes para la clasificación de libros, concretamente aquellas disponibles en red, con apoyo en la nube. Un cambio de objetivo tan drástico no hizo perder el progreso de la implementación, pero sí retrasó el comienzo efectivo del proyecto durante dos meses.

Además del proyecto, las tecnologías utilizadas han presentado diversos problemas. Comenzando por el que ha afectado directamente a los resultados de los experimentos, la base de datos generada por diferencia entre los sets balanceado y sin balancear ha creado un experimento de dudosos resultados, que ha hecho inútiles dos de los experimentos que podrían compararse con los resultados obtenidos por Microsoft. Sin embargo, el tratamiento de textos no ha sido el único problema, la librería TensorFlow recibió una actualización durante la etapa de pruebas del proyecto, y debido a problemas técnicos con el sistema operativo de Ubuntu 17.04 se requería una reinstalación de la librería. Desde aquel momento, TensorFlow dejó de funcionar como debiera, impidiendo la importación de módulos de Python necesarios para la ejecución del programa, a pesar de realizar todo el proceso de implementación en un entorno virtual.

Como problema de librerías, pero sobre todo de tiempo cabe destacar la exclusión de un segundo modelo implementado localmente. Además del Perceptrón, en la planificación del proyecto se incluía una red de neuronas recursiva utilizando LSTM (Long Short Term Memory), que en el campo de la clasificación de textos tiene gran relevancia por su capacidad de "recordar" relaciones entre distintas características de los datos. La librería derivada de TensorFlow, TFLearn, no era capaz de instalar los módulos requeridos en el ordenador de desarrollo, por lo que se abandonó la idea para poder continuar con el itinerario sin modificar las fechas establecidas.

6.4 Conclusiones personales

A título personal, este proyecto ha resultado ser una tarea más ardua de la que se visualizaba en primera instancia, nunca podría haber imaginado que los servicios de Machine Learning disponibles para el público general estarían tan avanzados, lo suficiente como para realizar una clasificación sobre un set de datos totalmente improvisado para el problema en sí. Ha sido un proyecto esclarecedor, que ha permitido que pudiese aprovechar mis habilidades de búsqueda de información, síntesis de la misma, análisis de resultados, planificación de pruebas, programación, diseño de redes neuronales, tratamiento de datos, etc.

He conseguido ampliar mi conocimiento con respecto al campo de las redes neuronales, que desde el comienzo del grado fue uno de los objetivos a cumplir, adentrarme más en el mundo de la Inteligencia Artificial. El aprendizaje automático fue el motor que me impulsó a estudiar esta carrera, y, tras concluir este proyecto, veo reafirmadas mis preferencias por esta materia.

Chapter 7: Feature extraction and Artificial Neural Networks for Classifying books by Literary Genre

7.1 Introduction

This section briefly summarizes the main points stated in the study explained above, using the English language to convey the most important information regarding the analysis, design, implementation and testing of the different parts that comprise this project.

The following sub-chapters contain key information to help understand the whole process of the study, distributed in a way that resembles the original work, omitting specific observations and focusing on just the main ideas extracted.

7.2 Main objectives

Nowadays all information produced by people has some degree of relevance for society. From financial transactions to comments written in social networks, everything is stored to be further analysed and serve as fuel for the machine that is knowledge. As such, literary works are another shape for information which people produce constantly, but it is of great importance to make good use of said information. To take advantage of this, the creation of algorithms and systems able to recognize patterns, to classify each example in a problem and – overall – learn from all this process to offer a better service.

The source of this study derives from the need of a classification system for a literary contest recommendation service for authors project. The role of the classifier would play is the validation of the submitted texts by the authors, making sure that the book or text provided by the user is indeed the genre stated in its submission. It also aims to classify those texts submitted without previous classification, in case it is needed. These functionalities would aid authors in finding suitable contests for their writing style and genre preference, and would protect the overall system from incorrect entries being made in wrong contests.

Aiming to obtain a good classifier, various Machine Learning systems were considered, mainly those offered by big technology companies which were available for public use, under fees in some cases. The possibilities in terms of complexity and extensibility for the networks designed by these companies promised better results than implementing manually known neural networks, which brings the point of the study: shed light over the capabilities of these cloud services for solving the proposed problem.

This comparative study aims primarily at understanding and checking whether a classification system exists among the main cloud based Machine Learning providers capable of discerning the literary genre of a book or other work with acceptable precision.

As this is a comparative study, another main objective is the implementation of a relatively simple neural network, locally developed, to establish a base precision that the cloud based systems must surpass.

It is worth noting that the study only aims for the classification of books and literary texts, taking only prose into account when classifying, thus excluding poetry and any form of lyric from the study.

7.3 System architecture

In this section the system environment and architecture will be discussed, describing the needed tools and elements for the design, implementation and testing of the networks used for the study.

First and foremost, this project aims to compare several systems or machine learning models, based in neural networks to obtain a classifier deemed precise enough to solve the classification problem described in the project objectives above. To this end, the project will study the behaviour of the neural networks designed ahead, including both the local neural network implemented manually and the cloud based classification systems provided by different companies. All possible candidates for cloud based classifiers will be discussed, although only the chosen for this project will be analysed in depth.

The study focuses solely on neural networks, and thus excludes from the comparison any model not related directly to this programming technique. It is known that other classifying models perform better in the task of classifying books or texts, but the purpose of this study is to prove the existence of cloud based neural network classifiers that fit the needs of the problem.

The system for developing the neural network and performing the evaluation of said network must follow these restrictions:

1. The Operative System (OS) where the neural net will be developed must be a Linux distribution, preferably from the Ubuntu family. Any version that supports Python 2.7 is required, but the usage of Ubuntu 16.04 or 17.04 is recommended.
2. The implementation language is Python 2.7.
3. The TensorFlow library used in the implementation must be of a version higher than 1.1.X.
4. A Python IDE is not required for the development of the model, and any text editor of choice may be used, only if it complies with Python' tab indentations.
5. A minimum storage space of 17 GB is needed to store both the model and the data.

As stated in these restrictions, the Multilayer Perceptron that will be used as a classifier for the project will be implemented in the Python library TensorFlow. This type of neural network was chosen among others due to its relative simplicity, as it must work only as a baseline for other models and should not consume too much time in terms of design, development, implementation and testing, as well as providing a base result for comparison, not the best possible solution.

Furthermore, the design, training and evaluation of the cloud based models will be carried out in a Windows 10 OS, using the latest update of the web browser Google Chrome.

7.3.1 Data processing

The database used in the project is a personal library from a collection of 10,193 books properly labelled and organized into 70 different classes. Due to uneven representation of all classes in the database, and the number of unique classes, the database was cut down to a set of unique classes of 20. The number of individual instances was thus reduced from 10,193 to 9,347 books.

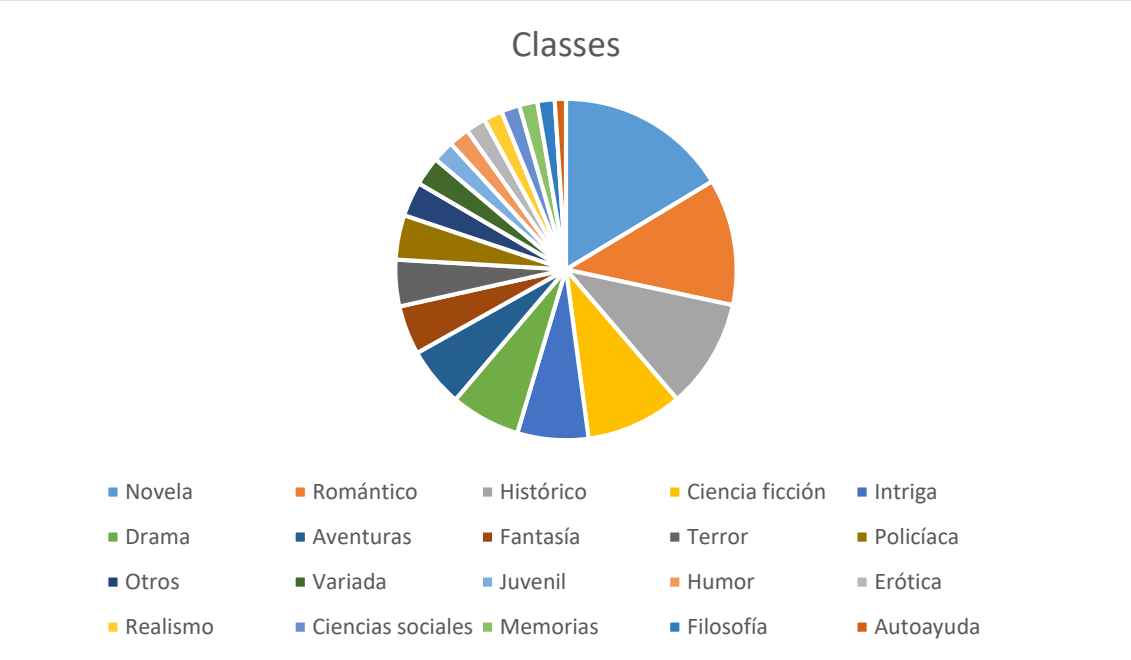


Figura 18: Representation of classes in 9,347 dataset

The individual tuples of the database were already randomly distributed as the base organization was in accordance to the book Author, but to further randomize the appearance of each class and distribute the most populated classes throughout the dataset a proper randomization was carried out.

A secondary database was made to further even the distributions of all the classes in the database. A threshold of 100 instances of each class (the minimum number of instances of the less populated class) was chosen for this training dataset, to even the learning of every class to avoid focusing on the better represented. The resulting database contains 2000 books.

After testing with the whole text as input for the models (previously parsing the text through a *stop word* removal process and special character elimination), since the workload was extremely big for every model it was decided that a change in the dataset was needed. Thus, the following datasets were created:

- 100 common words: Taking the texts from each book only the first 100 most frequent words were left in the set, making each individual word a feature for the models to train on.
- 100 common words balanced: This dataset uses the same data processing script as the first common words dataset, but uses as input the 2000 examples dataset to include only the balanced representation of the classes.
- 100 common words difference: A third dataset was created to validate the models created using the 100 common words balanced dataset. It contains the difference calculated from the first two new datasets described in this list.

With this new distribution of data, the new input would take only 100 words as input, and a single class for each example.

7.4 Network design and implementation

7.4.1 Multilayer Perceptron

The design of the Multilayer Perceptron (MLP) was obtained from a simple tutorial found online (dmesquita, 2017), in which the author describes the process of training and testing of a MLP for text classification, in their case for the classification of press articles. As it is a network designed for a similar task to this project's own, this model was selected as a good candidate for a baseline classification.

The only modifications made to the original design were carried out to accommodate the use of the database acquired for the project. Since the original design considered a total vocabulary different from the one used in this project, the input layer is smaller in the original design, but since the code only depends on the vocabulary used, the input of the correct vocabulary to make the *word2vec* implementation was sufficient. Moreover, the output of the network is also bigger than the original model, consisting of 20 different classes as stated in the database used.

The design of the MLP consists of an input and output layers as defined above, followed by two hidden layers of 100 neurons each, with a fully connected case. This design was left unchanged throughout the project, to ensure that the original design was unaltered when not necessary.

The input that the network takes is composed of an array of *n* positions, each representing a word of the total vocabulary among all books in the dataset. Each position contains the representation of the words within each example, thus providing the network with a frequency of each term in training example. Since the input dataset varies among full text and 100 words as input, this was changed matching the needs of each experiment carried out.

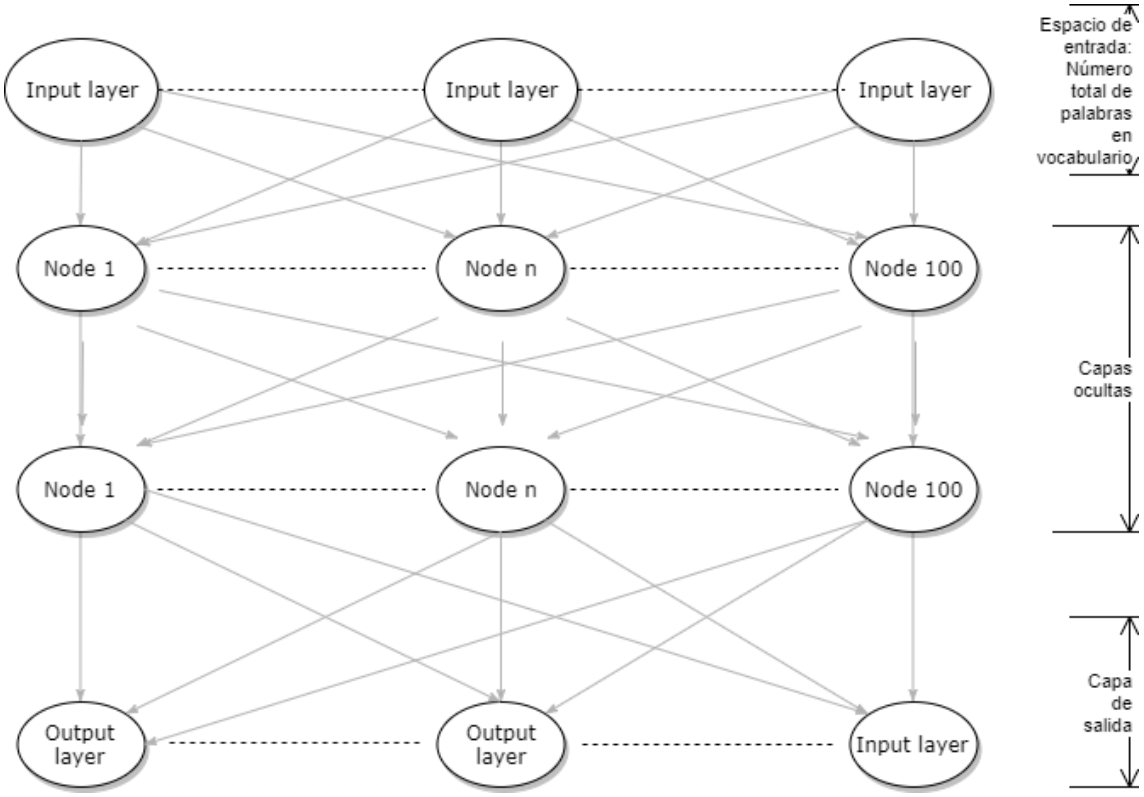


Figura 19: Multilayer Perceptron diagram

The loss function used in this model is the SoftMax function described in chapter 2. For further information, check the references section regarding SoftMax. The optimization function used in the study is the Adam algorithm, which can also be found in chapter 2, and further information is detailed by following the references regarding the Adam algorithm.

7.4.2 Cloud Machine Learning Systems

Regarding the cloud based systems studied, only 2 out of the initial 6 studied systems were useful for this project. The studied systems are: Google Cloud Machine Learning, Amazon Web Services Machine Learning, Microsoft Azure Machine Learning Studio, IBM Watson, BigML and SAP.

7.4.2.1 Google Cloud ML

Google Cloud ML was the first big company product to be evaluated, and had high hopes of becoming the best classifier among the studied solely from the reputation with Machine Learning that their company has. The products this company already developed for classification purposes are widely known: Cloud Vision can recognize a rather large set of elements from a picture using ML, Cloud Speech can translate to text a voice or audio file in real time, with astonishing results, and Google Translate is widely accepted as one of the best real time automated translators, with the help of ML. Of course, it would not be so easy, since upon further investigation it was discovered that Google did not offer a ML solution to be implemented via web browser or network editor, it only provides a Machine Learning API that can be accessed using their own Cloud Shell to request training jobs and evaluations of models trained on their platform. This would defeat the purpose of testing the ML solutions proposed by the companies, since the model to be trained would have to be built from scratch by the user. As such, Google Cloud was discarded from the list of potential systems to implement.

7.4.2.2 IBM Watson, BigML and SAP

IBM Watson ML will be a ML module to be added to the current IBM Watson development platform, but as of now it is impossible to use any of the features described in their “Coming Soon” page. There are implemented models that can be used from IBM Watson, but none of them can be adapted to the task to be solved in this project.

BigML is, like Google Cloud ML, an API that can be used to communicate local designs of networks or models to the servers of the company. Since the focus of the study is not to develop the same or different models in different APIs but rather compare the existing ones, this service is also discarded from the possible candidates.

SAP is a big company that services other companies mainly, offering products developed by them to solve common problems regarding Big Data analysis and processing. Although they can handle these tasks with ease, they don’t have any kind of classifier implemented that would benefit the study. This service was also not taken into consideration for the project.

7.4.2.3 Amazon Web Services

Amazon Web Services is a big platform that contains a variety of different services ranging from data storage online to Machine Learning engines that will allow their users to make their own models or use the default ones changing a few parameters to categorize or predict their desired data. This does align with the aim of the project, since it allows users to input their own data into the service, use this data to make a model out of the dataset provided, and designing a

neural network capable of classifying almost any kind of dataset, proving a fit candidate for our study.

Upon further investigating the tools offered by this cloud platform it was discovered that the process of creating a ML model was rather simple, giving the users any and every freedom to design their own neural network if they so choose. Given the objective of this study, the default network was used in the experiments, albeit changing some parameters to evaluate the changes in performance by tweaking the given models.

Regarding the model itself, only training parameters could be changed, and those will be covered in the next section. The actual distribution of the network is not known when training the model, but that would only be a hindrance to the project if it aimed to mimic the performance obtained by the cloud based model. But alas, that is not the case, and it can remain unknown as long as it is able to classify properly each instance of the dataset.

7.4.2.4 Microsoft Azure Machine Learning Studio

Microsoft also provides a big range of services in its Microsoft Cortana Intelligence Suite, but the aim of the project highlights the module named Azure Machine Learning Studio. This service provides an intuitive Machine Learning experiment builder, from data obtaining to model evaluation, it covers the entire model creation with virtually no limitations regarding the execution or design of each part. This system would prove the most useful were it not for the lack of more than one Neural Network model, at least for multiple class classification. As such, it lands together in terms of service offer with AWS.

Although the service is not as broad as one would want, not including Deep learning techniques other than a single class classifier, it provides a wide range of customization options for the users to carry out their own experiments with their data, or using the data provided by Microsoft itself.

The data processing service it provides is also worth noting, it prepares the data a user uploads so that it can be used in any experiment with little to no further modifications. It lets the user decide which datasets will be split for training and testing, and shows statistics on the data such as word count and frequency.

Regarding the models, the multiclass neural network classifier lets the user input their own design, but as in the case of AWS, the aim of the study is not to create a new model, but rather investigate which service can bring up the better model out of the data and parameters chosen. The design itself unaltered, we can tune the training parameters, which will be covered in the next section.

7.5 Training, testing and evaluation

7.5.1 Multilayer Perceptron

The training of the model was done by introducing as input the datasets described above, first using a small amount of texts to test the correct functionality of the model. After checking and testing each segment of the model performance using the tools provided by TensorFlow, the proper training was carried out. The results of each experiment were recorded in separate files, as well as saving the trained models for later validation. Each training set of data was composed of the 70% of the set, reserving the other 30% for the validation. Regarding the 100 common words with the balanced set of examples, the 2000 tuples were used for training and the difference set was used for validation. This is the only experiment where the training set is smaller than the validation set.

In each experiment, training values were tweaked to generate a variety of tests, and only the best ones were selected from among the total tests. The values that undergo variation are the number of epochs of training for each dataset and the learning rate of the model.

The trained models with the different datasets were validated using the 30% of the training sets which was reserved for this purpose. After validation, the model gave a measure of accuracy to evaluate the model regarding the loss function. The values obtained for each experiment where:

Dataset	Epochs,l.r.	Accuracy
Full text (preprocessed)	10,0.01	0.259
100 common words	25,0.01	0.347
100 common words balanced	10,0.1	0.316

These values of accuracy cannot be classified as good under any standards, since not even one of them reaches a 50% of accuracy when making predictions about validation data. Although this may be true, it is also not the worse it could be, since a baseline for this model would be a ZeroR classifier, which would get an accuracy of around 0.15 when choosing the most populated class *Novela*.

However, as it will be seen further ahead, the values of these experiments are too shallow to be compared with the cloud based accuracy measurements, due to being unable to access more data from each experiment, as the process of validation couldn't be analysed before the end of the project span.

7.5.2 AWS and MS Azure

For these two cloud based Machine Learning services several experiments were carried out. The decision to cut down the results of the experiments for the MLP classifier were due to the fact that both web services did so prior to showing the output data to the user, thus reducing the amount of "failed" tests carried out before the final model is created. They are alluded to as failed but the reality is that the systems remove these tests for not obtaining the highest score in their respective scoring process.

Regarding the results obtained from MS Azure, since this section is a summary of the whole testing process, it can be said that the accuracy of the models is not worthy of being called "good". Even though they achieve good performance for certain classes, there are several misclassifications in certain classes, that are frequent among the different experiments. Although these confusions are present, and are visible in any of the confusion matrices given as output by the service, there is room for improvement.

Observing the results of the Amazon Web Services tests, we can see that the classifiers using the 100 common words dataset perform very similarly to those in the MS Azure experiments. The same kind of mistakes are made, misclassifying classes as *Novela* with *Romántico*, and vice versa. The extensive analysis of these results lead to think that there is indeed a close relationship between these two classes, and makes it clearer that certain literary genres do indeed have close similarities when being classified as individual or unique.

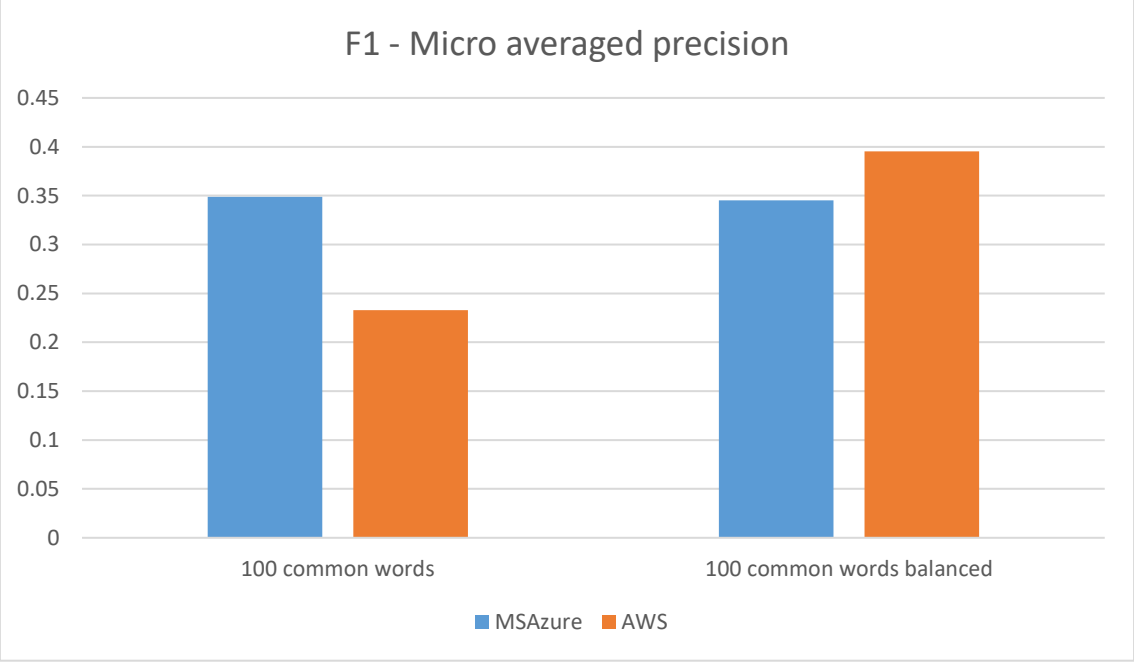


Gráfico 4: F1 - Micro averaged precision

Upon comparison of the accuracy measurements taken during testing (in the case of MS Azure, various measures of accuracy are taken, but the intersection of the two give the common function F1 as the micro averaged precision) we can see that Microsoft performs slightly better for the first dataset, while Amazon overperforms Microsoft in the balanced dataset. We cannot take this information as true however, since the results obtained by Amazon for the balanced dataset are suspiciously flawed. The output of the experiments carried out using this dataset reveal that there is a probability of repeated tuples existing in the training and validation sets.

Since these experiments were carried out using a larger validation set than the training set, it is no wonder that the results should be worse than normal. But it is quite the opposite in the case of the AWS experiments, since the results are far better than the MS counterpart.

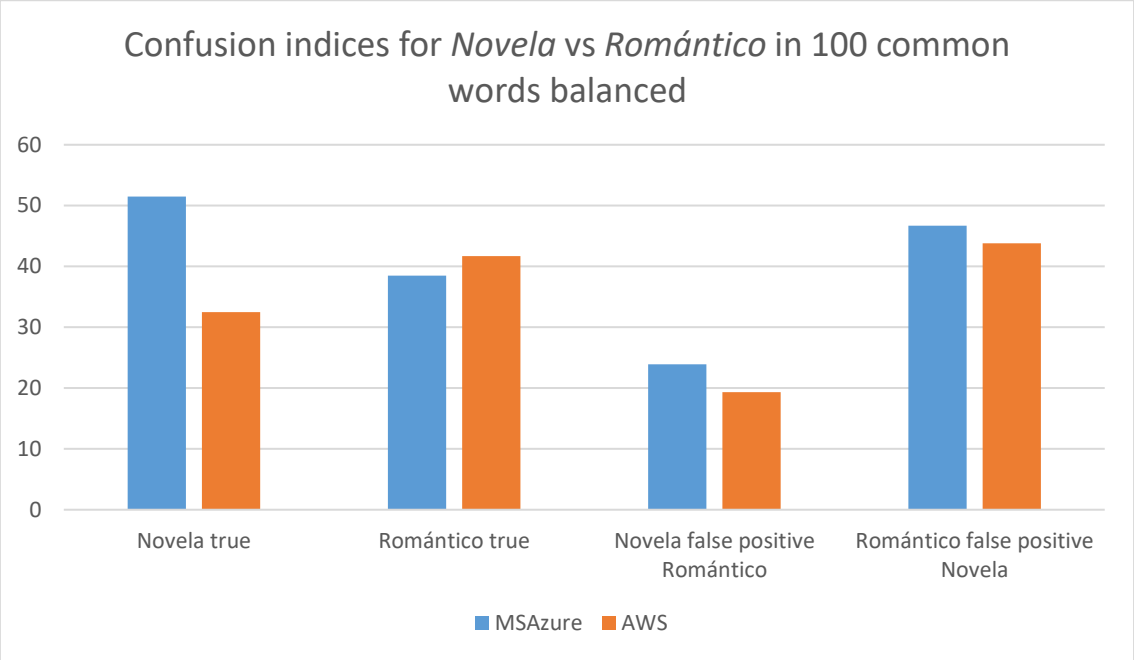


Gráfico 5: Confusion indices for Novela vs Romántico in 100 common words balanced

In the comparison of the above-mentioned relationship between the two classes analysed in the graph, we can see the close relationship both classifiers establish between the two classes, giving enough proof about the existing similarities between the literary genres.

These results broadly mean that a classifier capable of classifying a second class related to the first one to be classified would have an accuracy vastly superior in a system that allowed multiple classification of each text.

7.6 Conclusions and future work

As a conclusion for the study performed over the two cloud based solutions for Machine Learning and the comparison established with the Multilayer Perceptron, first the main objectives must be revised.

The objective regarding implementing a suitable Multilayer Perceptron to act as a baseline for the other classification systems is not fulfilled, since the comparison among the three classifiers studied is virtually impossible, due to the lack of information on the MLPs true performance, according to the F1 function.

Regarding the objective of finding a proper classifier among the cloud solutions offered by great companies, it can be said that the classifiers themselves performed in a good fashion, giving satisfying results that further deepen the understanding of the relations among different literary genres. Despite all this, the objective itself is only fulfilled if the system receiving the classifier is capable of accepting a second class, for what the classification models must be revised to accommodate this new feature.

Finally, with regards to future work in this field, further investigation upon the relationships among different literary genres would shed even more light to the similarities among them, and with the use of techniques like embedding in LSTM (Long Short Term Memory) Deep Neural Networks these features would become apparent with relative ease, according to the results of this study.

Referencias

Ayodele, T. O. (1 de February de 2010). *Types of Machine Learning Algorithms.* Obtenido de intechopen: https://www.intechopen.com/books/new-advances-in-machine-learning/types-of-machine-learning-algorithms

Chiang, H., Ge, Y., & Wu, C. (2015). *Classification of Book Genres By Cover and Title.* Stanford: Stanford University.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 303-314.

Davidian, D. (1995). *United States of America Patente nº US5438646 A.*

Diederik P. Kingma, J. B. (2015). Adam: A Method for Stochastic Optimization. *3rd International Conference for Learning Representations.* San Diego: Cornell University Library.

dmesquita. (19 de April de 2017). *Understanding TensorFlow.* Obtenido de Github: https://github.com/dmesquita/understanding_tensorflow_nn

Drakos, N., & Moore, R. (7 de April de 2009). *NLP*. Obtenido de Stemming and lemmatization: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research 15*, 3313-3181.

García-Pedrero, A., Gonzalo-Martín, C., & Lillo-Saavedra, M. (2017). A machine learning approach for agricultural parcel delineation through agglomerative segmentation. *International Journal of Remote Sensing*, 1809-1819.

Karpathy, A. (21 de May de 2015). *The Unreasonable Effectiveness of Recurrent Neural Networks*. Obtenido de Github: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Nielsen, M. (May de 2017). *Chapter 1 Using neural nets to recognize handwritten digits*. Obtenido de Neural Networks and Deep learning: http://neuralnetworksanddeeplearning.com/chap1.html

Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning, 39*, 103-134.

Pawar, P. Y., & Gawande, S. H. (2012). A Comparative Study on Different Types of Approaches to Text Categorization. *International Journal of Machine Learning and Computing, Vol. 2, No. 4*, 423-426.

Petrenz, P., & Webber, B. (2010). *Stable Classification of Text Genres.* Association for Computational Linguistics.

Porter, D. M. (s.f.). *Spanish stemming algorithm.* Obtenido de Snowball: http://snowballstem.org/algorithms/spanish/stemmer.html

Stergiou, C., & Siganos, D. (s.f.). *Neural Networks.* London: Imperial College.

UFLDL Standford. (s.f.). *Softmax Regression*. Obtenido de UFLDL: http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/

Vailant, L. G. (1984). A theory of the learnable. *Association for Computing Machinery*, 1134-1142.

Vijayarekha, D. K. (s.f.). *Activation Functions.* Obtenido de NPTEL: http://nptel.ac.in/courses/117106100/Module%208/Lecture%202/LECTURE%202.pdf